软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.005748]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 SVM 的多项式循环程序秩函数生成*

李 轶[1], 蔡天训[2], 樊建峰[1,3], 吴文渊[1], 冯 勇[1]

[1](中国科学院重庆绿色智能技术研究 自动推理与认知中心,重庆 400714)

[2](萨基姆通讯(深圳)有限公司,广东 深圳 518000)

[3](中国科学院大学,计算机与控制学院,北京 100093)

通讯作者: 李轶, E-mail: zm_liyi@163.com

摘 要: 程序终止性问题是自动程序验证领域中的一个研究热点. 秩函数探测是进行终止性分析的主要方法. 针对单重无条件分支的多项式循环程序,本文将其秩函数计算问题归结为二分类问题,从而可利用支持向量机(SVM)算法去计算程序的秩函数. 与基于量词消去技术的秩函数计算方法不同,本文方法能在可接受的时间内探测到更为复杂的秩函数.

关键词: 程序终止性;SVM;机器学习;秩函数

中图法分类号: TP311

中文引用格式:李轶,蔡天训,樊建峰,冯勇,吴文渊.基于 SVM 的多项式循环程序秩函数生成.软件学报,2019,30(7). http://www.jos.org.cn/1000-9825/5748.htm

英文引用格式:Li Y, Cai TX, Fan JF, Wu WY, Feng Y. An SVM-based method of detecting ranking functions for polynomial loop programs. Ruan Jian Xue Bao/Journal of Software, 2019,30(7) (in Chinese). http://www.jos.org.cn/1000-9825/5748.htm

SVM-Based Method for Detecting Ranking Functions in Polynomial Loop Programs

LI Yi[1], CAI Tian-Xun[2], FAN Jian-Feng[1,3], WU Wen-Yuan[1], FENG Yong[1]

[1](Chongqing Key Laboratory of Automated Reasoning and Cognition, CIGIT, CAS, Chongqing 400714, China)

[2](Sagemcom Technologies Inc, Shenzhen 518000, China)

[3](University of Chinese Academy of Sciences, Beijing 100093, China)

Abstract: Synthesizing ranking functions of polynomial loop programs is the dominant method for checking their termination. In this paper, the synthesis of ranking functions of a class of polynomial loop program is reduced to the binary problem. The support vector machine (SVM) technique then is applied to solve such the binary problem. This naturally relates detection of ranking functions to SVM. Different from the CAD-based method for synthesizing ranking functions, our method can get more expressive polynomial ranking functions in an acceptable time.

Key words: Program Termination, SVM, Machine learning, Ranking Functions

循环程序的终止性分析是程序验证的一个重要研究分支. 在现代程序设计中,几乎所有的程序都会含有循环. 然而,即使是简单的循环程序,也很容易出错. 循环中的很多错误往往需要执行多次或在某些特定情况下才能被发现. 因此,确保循环程序是可终止的是保障软件能够可靠运行的一个必要条件. 尽管程序终止性问题早已被证明是一个不可判定问题[1,2]. 但,人们发现具有某些特征的循环程序的终止性是可判定的. 如:Tiwari 在 2004 年证明了一类线性循环程序在实数域上的终止性是可判定[3]. 这类程序的终止性在文[4-7]中被重新考虑. 此外,Zhan 等人在[8]中考虑了循环条件为等式的多项式程序的终止性问题,并给出了该类程序

* 基金项目: 国家自然科学基金(61572024, 61103110,11471307);
Foundation item: National Natural Science Foundation of China (61572024, 61103110,11471307);
收稿时间: 2018-07-10; 修改时间: 2018-09-28; 采用时间: 2018-12-13; jos 在线出版时间: 2019-03-28
CNKI 网络优先出版: 2019-03-29 09:14:08, http://kns.cnki.net/kcms/detail/11.2560.TP.20190329.0913.001.html

可终止和不可终止的充分判准. 在文[9]中,Zhang 等人建立了针对程序终止性验证的高级自动机算法. 在循环非终止研究方面,Zhang 给出了一种能检测出简单循环中出现死循环的充分条件[10]. 针对确定型线性赋值循环程序,Leike 等人在文[11]建立了 GNTA 条件去探测这类循环的不可终止性.

秩函数法是证明循环程序可终止性的主要方法. 给定一个循环程序,倘若它的秩函数被找到,则表明该循环是终止的. 为计算秩函数方便,人们往往计算多项式型的秩函数. 根据多项式秩函数的次数可以将多项式秩函数分为线性秩函数和非线性多项式秩函数.

在线性秩函数研究方面,已有大量工作.譬如,2001 年,Colón 和 Sipma 通过凸多面体理论以及 Farkas' Lemma 给出了一种计算线性秩函数的方法[12,13]. 2004 年, Podelski 和 Rybalcheko 提出了一个完备的方法来构造线性约束的单分支循环程序的线性秩函数[14]. 2012 年,Bagnara 等人[15]研究了线性程序的线性秩函数的计算复杂度问题. 在 2013 年,他们提出了最终的线性秩函数(Eventual ranking functions)的概念[16],并通过合成最终的线性秩函数来证明单分支线性约束循环程序的终止性. 该方法首先利用一个单调增函数对循环程序的状态空间进行划分,然后利用 Farkas' Lemma 来合成最终的秩函数.文[17]将最终的线性秩函数进行了推广,提出了深度为 L 的最终线性秩函数. 2014 年, Leike 等人针对线性循环程序,给出了包括多阶段秩函数在内的多个秩函数定义[18]. 2017 年, Amir M. Ben-Amram 等人[19]对多阶段秩函数进行了进一步的研究,给出了一个能够在多项式时间内合成多阶段秩函数的方法.同时他们还指出了多阶段秩函数和线性字典序秩函数(Lexicographic Linear Ranking Functions)、嵌套秩函数(nested ranking function)之间的相互联系. 相较于线性秩函数的研究,非线性多项式秩函数合成方面的工作较少,主要有以下几篇文献讨论了非线性多项式秩函数的计算.Chen 等人[20]利用柱形代数分解(CAD)算法给出了一种合成非线性多项式秩函数的方法. 该方法首先设定循环程序的非线性秩函数模板,然后将秩函数合成问题转化为半代数系统求解问题,从而可利用符号计算工具 DISCOVERER 和 QEPCAD 来求解半代数系统,最终得出非线性秩函数模板中参数的解. 此外,文[21,22]等人通过利用半正定规划 SDP 工具去计算非线性多项式秩函数.

但程序的秩函数未必是多项式的. 也即,可以构造一个循环程序,它并没有多项式秩函数但具有其他形式秩函数[23]. 因此,当给定程序没有多项式秩函数时,我们还不能说它是不可终止的,因为它有可能有其他形式的秩函数. 在本文中,我们主要研究如下所示的单重无条件分支的循环程序,

$$\widetilde{P}\quad \begin{array}{l} \textit{while } \widetilde{\boldsymbol{p}}(x) \triangleright 0 \textit{ do} \\ \{x := \widetilde{F}(x)\} \\ \textit{endwhile} \end{array} \qquad (1)$$

其中 $x=(x_1,x_2,L,x_n)^T\in R^n$ 是循环程序的变量, $\widetilde{\boldsymbol{p}}(x)$ 为关于 x 的多项式向量,即 $\widetilde{\boldsymbol{p}}(x)=(\widetilde{p}_1(x),\ldots,\widetilde{p}_s(x))$,

其中 $\triangleright=(\triangleright_1,\ldots,\triangleright_m),\triangleright_i\in\{>,\geq\}$, $\widetilde{\boldsymbol{p}}(x)\triangleright 0$ 为循环程序 $\widetilde{P}$ 的循环条件, $x:=\widetilde{F}(x)$ 为赋值语句,其中,

$\widetilde{F}(x)=(\widetilde{f}(x),\widetilde{f}(x),\ldots,\widetilde{f}(x)), f_j(x)\in R[x]$. 既然程序中的所有表达式均是多项式的,因此程序 P 被称为多项式循环程序.令 $S=\{x:\widetilde{\boldsymbol{p}}(x)\triangleright 0\}$ 我们称程序 $\widetilde{\boldsymbol{p}}$ 是不可终止的,如果存在一点 $x^*\in R^n$,使得对任意的非负整数 k ,有 $\widetilde{\boldsymbol{p}}(x)(x^*)\in S$ 如果这样的点不存在,则称程序 $\widetilde{P}$ 在实数域上是可终止的.这里,

$\widetilde{F}^k=\widetilde{F}\circ\widetilde{F}\circ\cdots\circ\widetilde{F}$.一般地,程序 $\widetilde{P}$ 中的多项式表达式 $\widetilde{p}_i,\widetilde{f}_i$ 并非是关于 x 的齐次多项式. 但,通过增加新的变量 z ,我们可将非齐次多项式 $\widetilde{p}_i,\widetilde{f}_i$ 变为齐次多项式. 令 H_{n,d_i} 为所有 n 元齐 d_i 次实系数多项式的集合. 那么,易知对任给定的非齐次多项式程序 $\widetilde{P}$,其总可被转换为齐次多项式程序 P

$$P\quad \begin{array}{l} while\ \boldsymbol{p}(x,z) \rhd 0\ \ and\ \ z>0\ do \\ \quad \{x := F(x,z), z := z^d\} \\ endwhile \end{array}$$

这里, $\boldsymbol{p}(x,z)=(p_1(x,z),\ldots,p_s(x,z))$, $F(x,z)=(f_1(x,z),\ldots,f_n(x,z))$,且对任意的 $i\in[1,s]$, $p_i(x,z)\in H_{n+1,d_i}$,对任意的 $j\in[1,n]$, $f_i(x,z)\in H_{n+1,d}$. 这里,对任意两个整数 a,b 且 $a\le$ b ,记 [a,b]={a,a+1,…,b-1,b} 由文[3,22]中的定理可知,程序 $\widetilde{P}$ 与程序 P 在终止性上是等价的.

因此,不失一般性,本文主要分析下列的齐次多项式循环程序终止性.

$$Q\quad \begin{array}{l} while\ \boldsymbol{C}(\boldsymbol{x}) \rhd 0\ \ do \\ \quad \{x := M(x)\} \\ endwhile \end{array}$$

这里, $x\in R^n$, $C(x)=(c_1(x),\ldots,c_s(x))$, $M(x)=(m_1(x),\ldots,m_n(x))$ 且对任意的 $i\in[1,s]$, $c_j(x)\in H_{n,d_i}$, 对任意的 $j\in[1,n], m_j(x)\in H_{n,d}$, $\rhd=(\rhd_1,\ldots,\rhd_s), \rhd_i\in\{>,\ge\}$,且必须存在某个 j ,使得 $\rhd_j \triangleq >$.也即,程序 Q 中,其循环条件中的每个多项式都是齐次的且次数未必均相同,此外至少一个不等式是严格的不等式. 同时程序 Q 的迭代映射 $M(x)$ 中的各个分量均是齐次多项式且具有相同次数. 既然 $C(x)$ 的分量均是齐次多项式,故有：若 $\boldsymbol{C}(\boldsymbol{x})\rhd 0$,则对任意正数 $\delta>0$ 有 $\boldsymbol{C}(\delta\boldsymbol{x})\rhd 0$.程序 Q 能被其对应的变迁系统刻画：

$\Delta_Q \triangleq \boldsymbol{C}(\boldsymbol{x})\rhd 0\ \wedge\ x'=M(x)$.其中, x 表示当前状态, x' 表示下一个状态.不失一般性,下文中,我们提到的程序 Q 均是指其对应的变迁系统. 令 $\Omega=\{x\in\mathbb{R}^n:\ \boldsymbol{C}(\boldsymbol{x})\rhd 0\}$.根据上述程序 Q 的定义,既然 $\boldsymbol{C}(\boldsymbol{x})\rhd 0$ 中至少存在一个严格不等式,那么原点 $O\notin\Omega$.对程序 Q ,秩函数的存在表明程序 Q 必定是终止的.

在下文中,我们将建立新的方法去探测程序 Q 的秩函数.该方法的主要思路是：将秩函数计算问题归结为二分类问题,然后利用机器学习中的支持向量机(SVM)算法[24-25]来合成所需秩函数,最终验证合成的秩函数是否满足秩函数定义.相较于当前的基于CAD或SDP的多项式秩函数计算方法,本文的主要贡献在于提出的基于 SVM 的方法用于计算非多项式型秩函数——代数分式型秩函数,从而将秩函数的研究从多项式型秩函数扩展到代数分式型秩函数,扩大了可计算秩函数类型的范围.

文章组织结构：第一部分介绍秩函数,支持向量机等相关概念;第二部分建立基于 SVM 的秩函数探测算法,并通过实例详细阐述该算法.第三部分给出更多实验.第四部分总结全文.

1 预备知识

1.1 秩函数

定义 1 记号同上. 给定齐次多项式循环程序 Q . 令 $\rho(x)$ 为一 n 元函数. $\rho(x)$ 被称为程序 Q 的秩函数,如果下列条件被满足：

(a) $\forall x.\boldsymbol{C}(\boldsymbol{x})\rhd 0\Rightarrow\rho(x)\ge d$ (d 为常数)

(b) $\forall x\forall x'.\boldsymbol{C}(\boldsymbol{x})\rhd 0\ \wedge\ x'=M(x)\Rightarrow\rho(x)-\rho(x')\ge 1$

上述条 (b) 能被简化为： $\forall x.\boldsymbol{C}(\boldsymbol{x})\rhd 0\Rightarrow\rho(x)-\rho(M(x))\ge 1$ 因此,等价地,如果 $\rho(x)$ 满足下列条件

(a) $\forall x.\boldsymbol{C}(\boldsymbol{x})\rhd 0\Rightarrow\rho(x)\ge d$

(c) $\forall x.\boldsymbol{C}(\boldsymbol{x})\rhd 0\Rightarrow\rho(x)-\rho(M(x))\ge 1$

则 $\rho(x)$ 为程序 Q 的秩函数.

1.2 支持向量机

支持向量机是一个性能良好的有监督的学习算法,其核心思想是通过构造超平面将数据进行分离.给定一组训练样本集 $D=\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_m,y_m)\},y_i\in\{-1,1\}$ 其中 $\boldsymbol{x}_1$ 是样本的属性, y_1 是对应样本的标签.如果存在某个超平面将不同标签的样本点分开我们就称该训练样本集是线性可分的.在训练样本集是线性可分的情况下,能够将训练样本集分开的超平面有许多,而我们的目的是寻找到泛化能力最强的那一个超平面.如图 1 所示,存在着多条超平面将图中的样本点分开,但我们需要找到的是中间那条线也就是最"居中"的那一个.

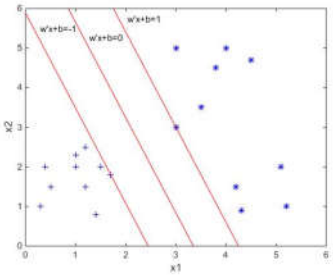

Fig.1 support vector and margin

图 1 支持向量与间隔

在样本空间中,划分超平面可用线性方程 $\boldsymbol{a}^T x+k=0$ 来描述,其中 $\boldsymbol{a}=(a_1,\ldots,a_d)$ 是超平面的法向量, k 是截距项.支持向量机可以用于找到最"居中"的那一个超平面.如图 1 所示,支持向量机可以找到划分超平面 $\boldsymbol{a}^T x+k=0$,该超平面将图中的两类样本点划分开,其中落在 $\boldsymbol{a}^T x+k=1$ 和 $\boldsymbol{a}^T x+k=-1$ 上的点叫做支持向量.支持向量机的求解通常使用 SMO 算法[24]来求解.LIBSVM[25]是一个关于 SVM 的集成软件包,由台湾大学的 Chih-Wei Hsu、Chih-Chung Chang,和 Chih-Jen Lin 编写完成.软件的主要部分采用 C 语言编写完成,可在 MATLAB 上调用.本文的主要计算部分采用该软件包完成.

2 主要结果

本节将阐述如何将型如程序 Q 的秩函数计算问题归结为二分类问题,以便可利用支持向量机(SVM)工具去计算得到一个候选秩函数,最后再对该候选秩函数进行验证,以检验其是否满足秩函数的两个条件 (a) 和 (c) .

一般地,计算程序 Q 的秩函数,需要首先预设一个秩函数模板.秩函数模板形式可以是多种多样的.但为方便计算,人们常将秩函数模板设定为多项式模板.但,由于程序 Q 是齐次多项式程序,即它中的所有表达式都是齐次多项式的,因此下面的定理告诉我们程序 Q 没有多项式秩函数.

定理 2 记号同上. 程序 Q 没有多项式秩函数.

证明: 假设程序 Q 具有多项式秩函数. 那么,不失一般性,设多项式 $\rho(x)=\sum_{C_\alpha}C_\alpha x^\alpha$ 为程序 Q 的秩函数模板. 这里, $\alpha=(\alpha_1,\ldots,\alpha_n),\alpha_i$ 为非负整数, $x^\alpha=x_1^{\alpha_1}\cdot x_2^{\alpha_2}\cdots x_n^{\alpha_n}$, C_α 为单项式 x^α 前的系数. 既然 $\rho(x)$ 为程序 Q 秩函数,根据秩函数定义 1, $\rho(x)$ 必然满足条件 (a) 和 (c) :

$$\forall x\boldsymbol{C}(\boldsymbol{x})\triangleright 0\Rightarrow\rho(x)=\sum_{C_\alpha}C_\alpha x^\alpha\geq d \qquad (2)$$

$$\forall x.\boldsymbol{C}(\boldsymbol{x})\triangleright 0\Rightarrow\rho(x)-\rho(M(x))=\sum_{C_\alpha}C_\alpha x^\alpha-\sum_{C_\alpha}C_\alpha M(x)^\alpha=\sum_{C_\alpha}C_\alpha(x^\alpha-M(x)^\alpha)\geq 1 \qquad (3)$$

这里, $M(x)^\alpha=m_1(x)^{\alpha_1}\cdot m_2(x)^{\alpha_2}\cdots m_n(x)^{\alpha_n}$. 因为 $\boldsymbol{C}(\boldsymbol{x})$ 的所有多项式都是齐次的,那么有

$$\forall x\forall\delta.\boldsymbol{C}(\boldsymbol{x})\rhd 0\ \wedge\ \delta>0\Rightarrow \boldsymbol{C}(\delta\boldsymbol{x})\rhd 0 \qquad (4)$$

任取 x^* 满足 $\boldsymbol{C}(x^*)\rhd 0$，由(4)可知,对任意的正数 δ,均有 $\boldsymbol{C}(\delta x^*)\rhd 0$. 那么有,当 $\delta\to 0$,

$$\sum_{C_\alpha} C_\alpha(\delta x^{*\alpha}-M(\delta x^*)^\alpha)\to 0\not\geq 1.$$

这显然与式(3)矛盾. 故程序 Q 没有多项 1 式秩函数. ■

上述定理表明程序 Q 不可能存在多项式秩函数,因此在计算其秩函数时,不能将其秩函数模板设置为多项式形式. 下文中,我们将表明程序 Q 的秩函数模板可以设置为代数分式型模板. 这里,代数分式是两个代数表达式的商,如 $\dfrac{x-y}{xy^4+x^2-1},\dfrac{\sqrt{xy}}{x+3y+2}$ 均是代数分式.

2.1 程序 Q 的秩函数计算归结为二分类问题

既然程序 Q 已被证明不可能有多项式型的秩函数,那么本节将讨论如何将程序 Q 的秩函数计算问题归结为二分类问题,从而便于利用支持向量机(SVM)算法去探测程序 Q 的代数分式型秩函数. 首先给出一些必要的记号. 记 $\Upsilon(x)=(x^{\alpha_1},x^{\alpha_2},\ldots,x^{\alpha_k})$.这里 $\Upsilon(x)$ 中的每个分量 x^{α_k} 为关于 x 的单项式,且 α_k 为非负整数向量. 并记 $|\alpha_k|$ 为单项式 x^{α_k} 的次数. 如 $x=(x_1,x_2),\alpha_k=(1,2)$,则 $x^{\alpha_k}=x_1^1x_2^2$. Υ 可被看作是 $\mathbb{R}^n\to\mathbb{R}^k$ 单项式 P 映射. 不失一般性,我们可以按次数将 γ 中的单项式分类排列,即

$$\Upsilon(x)=(x^{\alpha_{i_1}},x^{\alpha_{i_2}},\ldots,x^{\alpha_{i_{j_1}}},x^{\alpha_{i_{j_1+1}}},\ldots,x^{\alpha_{i_{j_2}}},x^{\alpha_{i_{j_2+1}}},\ldots,x^{\alpha_{i_{j_3}}},\ldots,\ldots,x^{\alpha_{i_{j_v+1}}}\ldots,x^{\alpha_{i_k}}).$$

这里,向量 γ 被分割成 $v+1$ 段. 其中,第一段 $\Upsilon_1(x)=(x^{\alpha_{i_1}},x^{\alpha_{i_2}},\ldots,x^{\alpha_{i_{j_1}}})$ 具有相同的次数,即 $|\alpha_{i_1}|=\cdots=|\alpha_{i_{j_1}}|$;第二段 $\Upsilon_2(x)=(x^{\alpha_{i_{j_1+1}}},\ldots,x^{\alpha_{i_{j_2}}})$ 具有相同的次数,即 $|\alpha_{i_{j_1+1}}|=\cdots=|\alpha_{i_{j_2}}|$,..., 第 $v+1$ 段 $\Upsilon_{v+1}(x)=(x^{\alpha_{i_{j_v+1}}},\ldots,x^{\alpha_{i_{j_k}}})$ 具有相同的次数,即 $|\alpha_{i_{j_v+1}}|=\cdots=|\alpha_{i_{j_k}}|$.令 $\sqrt{\sum(\Upsilon_v(x)^2)}$ 表示将第 v 段 $\Upsilon_v(x)$ 中的所有分量各自平方后相加再开根号. 令

$$T(x)=\left(\frac{\Upsilon_1(x)}{\sqrt{\sum(\Upsilon_1(x)^2)}},\frac{\Upsilon_2(x)}{\sqrt{\sum(\Upsilon_2(x)^2)}},\cdots,\frac{\Upsilon_{v+1}(x)}{\sqrt{\sum(\Upsilon_{v+1}(x)^2)}}\right).$$

显然, $T(x)$ 可被看作是 $\mathbb{R}^n\to\mathbb{R}^k$ 映射. 并且, $T(x)$ 各个分量的分子分母齐次且次数相同,故对任意的 $\lambda>0$ 有 $T(x)=T(\lambda x)$.令

$$U(x)=T(x)-T(M(x))=\left(\begin{array}{l}\dfrac{\Upsilon_1(x)}{\sqrt{\sum(\Upsilon_1(x)^2)}}-\dfrac{\Upsilon_1(M(x))}{\sqrt{\sum(\Upsilon_1(M(x))^2)}},\dfrac{\Upsilon_2(x)}{\sqrt{\sum(\Upsilon_2(x)^2)}}-\dfrac{\Upsilon_2(M(x))}{\sqrt{\sum(\Upsilon_2(M(x))^2)}},\\ \cdots,\dfrac{\Upsilon_{v+1}(x)}{\sqrt{\sum(\Upsilon_{v+1}(x)^2)}}-\dfrac{\Upsilon_{v+1}(M(x))}{\sqrt{\sum(\Upsilon_{v+1}(M(x))^2)}}\end{array}\right)$$

. 同样, $U(x)$ 可被看作是 $\mathbb{R}^n\to\mathbb{R}^k$ 映射并且由于 $U(x)$ 的特殊构造型式,使得对任意的 $\lambda>0$ 有 $U(x)=U(\lambda x)$ 即在同一条射线上的任意两非零点 $x,\lambda x$ 它们在函数 $U(x)$ 的映射下的像是相同的.

下面的命题表明,若 $T(x)$ 在 Ω 上有定义,那么,函数 $T(x)$ 为有界函数.也即,存在正数 δ ,使得对任意的 $x\in\Omega$ 有 $|T(x)|\leq\delta$.

命题 3 记号同上. 若 $T(x)$ 在 Ω 上有定义,那么, $T(x)$ 在 Ω 上有界.

证明：既然 $T(x)$ 在 Ω 上有定义,则对 Ω 中任意一点 x , $T(x)$ 中各个分量的分母均不会为 0. 考虑 $T(x)$ 中第 v 段,即 $\dfrac{\Upsilon_v(x)}{\sqrt{\sum(\Upsilon_v(x)^2)}}=\left(\dfrac{x^{\alpha_{i_{j_{v-1}+1}}}}{\sqrt{\sum(\Upsilon_v(x)^2)}},\cdots,\dfrac{x^{\alpha_{i_{j_v}}}}{\sqrt{\sum(\Upsilon_v(x)^2)}}\right)$. 这里,

$$\sqrt{\sum(\Upsilon_v(x)^2)}=\sqrt{(x^{\alpha_{i_{j_{v-1}+1}}})^2+\cdots+(x^{\alpha_{i_{j_v}}})^2}\ .$$

显然有,

$$\left|\frac{\Upsilon_v(x)}{\sqrt{\sum(\Upsilon_v(x)^2)}}\right|=\left|\left(\frac{x^{\alpha_{i_{j_{v-1}+1}}}}{\sqrt{\sum(\Upsilon_v(x)^2)}},\cdots,\frac{x^{\alpha_{i_{j_v}}}}{\sqrt{\sum(\Upsilon_v(x)^2)}}\right)\right|$$

$$\le\left|\frac{x^{\alpha_{i_{j_{v-1}+1}}}}{\sqrt{\sum(\Upsilon_v(x)^2)}}\right|+\cdots+\left|\frac{x^{\alpha_{i_{j_v}}}}{\sqrt{\sum(\Upsilon_v(x)^2)}}\right|\le 1+\cdots+1\le j_v-j_{v-1}<\infty$$

因此, $\dfrac{\Upsilon_v(x)}{\sqrt{\sum(\Upsilon_v(x)^2)}}$ 在 Ω 上有界. 既然 $T(x)$ 中的每一段都是有界的,故 $T(x)$ 在 Ω 上有界. □

既然 $T(x),U(x)$ 可被看作是 $\mathbb{R}^n\to\mathbb{R}^k$ 映射,若它们在 Ω 上有定义,则记

$$T(\Omega)=\{t\in\mathbb{R}^k:t=T(x),x\in\Omega\},U(\Omega)=\{u\in\mathbb{R}^k:u=U(x),x\in\Omega\}$$

根据命题 3,若 $T(x)$ 在 Ω 上有定义,那么 $T(\Omega)$ 是 $\mathbb{R}^k$ 空间中的有界域. 倘若存在某 $a\in\mathbb{R}^k$,使得连续函数 $L(u)=a^T\cdot u$ 在 $U(\Omega)$ 上有下界1, 即

$$\forall u\in U(\Omega)\Rightarrow L(u)=a^T\cdot u\ge 1 \tag{5}$$

那么,根据 $U(\Omega)$ 的定义有,

$$\forall x\in\Omega\Rightarrow a^T\cdot(T(x)-T(M(x)))\ge 1 \tag{6}$$

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 $\forall t\in T(\Omega)\Rightarrow L'(t)=a^T\cdot t\ge c$.根据 $T(\Omega)$ 的定义, 上式等价于

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

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(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,我们可以通过去计算能满足式(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 $\forall u\in U(\Omega)\Rightarrow L(u)=a^T\cdot u\ge 1$,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 的象集 $U(\Omega)$ 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 $a^T \cdot O \equiv 0 \not\geq 1$. 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 , $a^T \cdot (T(x)-T(M(x))) \equiv 0 \not\geq 1$. 而程序 Q 被给定时,象集 $U(\Omega)$ 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:

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

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

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

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

(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),而(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 $\pi(O) < 0$.那么有 $\pi(O) = 0 + k = k < 0$.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 等价于寻找能将原点 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,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$\Omega$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$\mathbb{R}^k$空间中能严格分离原点$O$与某个包含$U(\Omega)$的闭集的严格分离超平面$\pi(u)$的问题. 为此首先给出一些如下记号.

令$\overline{\Omega}=\{x\in\mathbb{R}^n:\boldsymbol{C}(\boldsymbol{x})\geq 0\},\Omega^*=\overline{\Omega}\setminus\{O\}$ 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,根据此前分析知,对任意的$\lambda>0$,有$U(x)=U(\lambda x)$. 这表明同一条射线上的任意两非零点在映射$U(x)$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$O\notin\Omega^*$,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$\frac{x}{|x|}\in\Omega^*\bigcap\Theta$. 因此$U(x)=U(\frac{x}{|x|})$,故而有$U(\Omega^*)=U(\Omega^*\bigcap\Theta)$. 此外，因为$\overline{\Omega}\bigcap\Theta=(\Omega^*\bigcup\{O\})\bigcap\Theta=\Omega^*\bigcap\Theta$.显然,因为$\overline{\Omega}$是闭的,$\Theta$是有界闭的,故$\overline{\Omega}\bigcap\Theta$也是有界闭的.由上式可知$\Omega^*\bigcap\Theta$也是有界闭的. 既然前面已经假设$U(x)$在$\Omega^*$上有定义,因此$U(x)$在$\Omega^*$上连续.有连续函数性质可知,$\Omega^*\bigcap\Theta$在$U(x)$的像$U(\Omega^*\bigcap\Theta)$是有界闭的. 既然$U(\Omega^*)=U(\Omega^*\bigcap\Theta)$，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$\forall u\in U(\Omega)\Rightarrow\frac{a^T}{-k}\cdot u\geq 1$. □

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)算法去解决. 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算法去求得一个能将集合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 算法去计算 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 算法去计算单点集 A 与有限点集B_0的分离超平面, 记为 $\pi_0(u) = a^T \cdot u + k$. 转 S3;

S3：验证式(5)：$\forall u \in U(\Omega) \Rightarrow \pi_0(u) = \dfrac{a^T}{-k} \cdot u \geq 1$是否成立. 若成立,则表明程序 Q 有代数分式型秩函数,即 $\rho(x) = \dfrac{a^T}{-k} \cdot T(x)$.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,我们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具体介绍. 在 S2 中,当调用 SVM 算法去计算单点集 A 与有限点集 B_0 的分离超平面 $\pi_0(u)$ 时,在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： $\forall x \in \Omega \Rightarrow \dfrac{a^T}{-k} \cdot (T(x) - T(M(x))) \geq 1$. 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 去验证上式是否成立. 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去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, $U(\Omega^*) = U(\Omega^* \bigcap \Theta)$. 这里, $\Omega^* \bigcap \Theta$ 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 $\Omega^* \bigcap \Theta$ 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 $B_0 = U(C)$. 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 $\Omega^* \bigcap \Theta$ 上的点集 C ,我们采取如下思路：先在区域 $\Omega^* \bigcap \Delta$ 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 $S^0 \subseteq \Omega^* \bigcap \Theta$.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 $\Omega^* \bigcap \Delta$ 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 .下面是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 $x^0 = \dfrac{x}{|x|}$ 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

$$Q_1 \quad \text{while} \quad 4x_1^2 + x_2^2 + 16 \le 16x_1 + 6x_2 \wedge x_1 + 4x_2 \ge x_2^2 + 5 \quad \text{do}$$

$$\{x'_1 = x_1^2 - x_2^2; x'_2 = x_1 + x_2 + 1;\}$$

该循环若使用 RegularChains[27]或者 redlog[28]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

$$Q_1' \quad \text{while} \quad 4x_1^2 + x_2^2 + 16x_3^2 \le 16x_1x_3 + 6x_2x_3 \wedge x_1x_3 + 4x_2x_3 \ge x_2^2 + 5x_3^2 \wedge x_3 > 0 \quad \text{do}$$

$$\{x'_1 = x_1^2 - x_2^2; x'_2 = x_1x_3 + x_2x_3 + x_3^2; x'_3 = x_3^2;\}$$

其中 $\Omega = \{x:\ 4x_1^2 + x_2^2 + 16x_3^2 \le 16x_1x_3 + 6x_2x_3 \wedge x_1x_3 + 4x_2x_3 \ge x_2^2 + 5x_3^2 \wedge x_3 > 0\}$ 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,也即是针对任意的 $x \in \Omega$,都有 $\lambda x \in \Omega, \lambda > 0$.令

$\bar{\Omega} = \{x:\ 4x_1^2 + x_2^2 + 16x_3^2 \le 16x_1x_3 + 6x_2x_3 \wedge x_1x_3 + 4x_2x_3 \ge x_2^2 + 5x_3^2 \wedge x_3 > 0\}$.令 $\Omega^* = \bar{\Omega} \setminus \{O\}$.这里, O 为原点 $(0,0,0)$.对该例,我们可令 $T(x) = (\frac{x_1}{|x|}, \frac{x_2}{|x|}, \frac{x_3}{|x|})$. $T(x)$ 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 $T(x') = (\frac{x'_1}{|x'|}, \frac{x'_2}{|x'|}, \frac{x'_3}{|x'|})$.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, $|x'| = \sqrt{(x'_1)^2 + (x'_2)^2 + (x'_3)^2}$. 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,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：

$$\begin{aligned}&\frac{x_1}{|x|}-\frac{x'_1}{|x'|}=0\wedge\frac{x_2}{|x|}-\frac{x'_2}{|x'|}=0\wedge\frac{x_3}{|x|}-\frac{x'_3}{|x'|}=0\wedge x'_1=x_1^2-x_2^2\wedge x'_2=x_1x_3+x_2x_3+x_3^2\wedge x'_3=x_3^2\\&\wedge 4x_1^2+x_2^2+16x_3^2\le 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\ge x_2^2+5x_3^2\wedge x_3\ge 0\\&\wedge(x_1\neq 0\vee x_2\neq 0\vee x_3\neq 0)\wedge|x'|=\sqrt{(x'_1)^2+(x'_2)^2+(x'_3)^2}\end{aligned}\tag{8}$$

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. 由此我们得到下列的多项式系统.

$$\begin{aligned}&x_1^2((x'_1)^2+(x'_2)^2+(x'_3)^2)=(x'_1)^2(x_1^2+x_2^2+x_3^2)\wedge x_2^2((x'_1)^2+(x'_2)^2+(x'_3)^2)=(x'_2)^2(x_1^2+x_2^2+x_3^2)\wedge\\&x_3^2((x'_1)^2+(x'_2)^2+(x'_3)^2)=(x'_3)^2(x_1^2+x_2^2+x_3^2)\wedge x'_1=x_1^2-x_2^2\wedge x'_2=x_1x_3+x_2x_3+x_3^2\wedge x'_3=x_3^2\wedge\\&4x_1^2+x_2^2+16x_3^2\le 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\ge x_2^2+5x_3^2\wedge x_3\ge 0\end{aligned}\tag{9}$$

显然,式(8)的任何一个解也必是式(9)的解. 调用 Maple 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 2.2 小节介绍的取点算法. 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 $\overline{\Omega}$ 约束条件中有 $x_3\ge 0$ 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 算法去计算 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 软件包的方式去计算一个“硬分隔”的超平面. 令 $\overline{D}=A\bigcup B_0$.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,

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

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 $\pi_0(O)<0$ ， 或者,

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

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成立：

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

(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 $U(\Omega^*)$ 的定义知 $U(\Omega^*) = \{u \in \mathbb{R}^k : u = U(x), x \in \Omega^*\}$,验证上述(i),(ii)两式等价于验证：

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

(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. 由于我们最终需要验证式(5)或式(6)是否成立,所以,在本例中,我们这里直接验证

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

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,我们发现

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

的确成立. 故,

$$\rho(x) = \frac{a^T}{-k} \cdot T(x) = \frac{(2.798739925884132, -6.058645127373750, 6.167554957795089)^T}{-1.000000000061875} \left(\frac{x_1}{|x|}, \frac{x_2}{|x|}, \frac{x_3}{|x|} \right)$$

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.

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 $\overline{\Omega}$.令 Ω*=$\overline{\Omega}\setminus\{O\}$. 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 $\overline{D} = \{O\} \bigcup B_0$ 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

实验是在一台装有 7.86GB 的可用内存,处理器频率为 2.59GHz,操作系统是 64 位的 windows 操作系统的计算机上进行的.下面我们选取了四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.

Table 1 More Examples

表 1 更多实例

P_2: while $(y^2+10 \le x+6y \wedge x^2+6 \le 4x+y)$

$$\begin{cases} x_1 = x+y; \\ y_1 = y-1; \end{cases}$$

P_3: while $(x^2+y^2 \le 1)$

$$\begin{cases} x_1 = x-y^2+1; \\ y_1 = y+x^2-1; \end{cases}$$

P_4: while $(x-y \ge 1 \wedge x+y \ge 1 \wedge x \le 10)$

$$\begin{cases} x_1 = y; \\ y_1 = y-1; \end{cases}$$

P_5: while $(x \ge 0 \wedge y \le -1)$

$$\begin{cases} x_1 = x+y; \\ y_1 = y-1; \end{cases}$$

P_6: while $(4 \le x \le 5 \wedge 1 \le y \le 2)$

$$\begin{cases} x_1 = x; \\ y_1 = -xy+y^2+1; \end{cases}$$

P_7: while $(x \ge 0 \wedge y-2x \ge 1)$

$$\begin{cases} x_1 = -x^2-4y^2+1; \\ y_1 = -xy-1; \end{cases}$$

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(53.9 分钟)

P_3	unknown	unknown	Terminating(79.5 分钟)
P_4	Terminating(0.24 秒)	Terminating(125 分钟)	Terminating(74.3 分钟)
P_5	Terminating(0.2 秒)	Terminating(6.156 秒)	Terminating(41.7 分钟)
P_6	unknown	unknown	Terminating(2.7 分钟)
P_7	unknown	unknown	Terminating(1.5 分钟)

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 中可以看出除了循环程序 P_4 和 P_5 外,其他五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”表示在计算时间超过 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,则基于量词消去的工具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,本文的提出的基于 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的时间.在实验中我们发现,本文方法最主要的时间开销是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探测出上述六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 所示.

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]	-0.999999997404251
P_3	0.5	[-5.303998785021788,5.303967565999045, 4.733548642017029]	-1.000213463236426
P_4	0.5	[114.542235487223684,136.417264854622173, -199.707636874516821]	-0.999859927713084
P_5	0.5	[82.9547,194.7744,-1.8830]	-0.9998
P_6	0.01	[-0.4922,5.1377,2.5196]	-1
P_7	0.5	[0.9720,1.0254,0.2133]	-0.9999

从表 3 中可以看出,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越高. 但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做适当的调整.

4 总结

本文提出了一个基于支持向量机(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去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,传统的量词消去的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, Front. Comput, Sci, 2010, 4(1):1-16

[3] Tiwari A. Termination of Linear Programs. Computer Aided Verification. Springer Berlin Heideiberg, 2004,3114, 70-82.

[4] Braverman M. Termination of integer linear programs. In: Ball, T., Jones, R.B.(eds.) Computer Aided Verification (CAV 2006), Springer, 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 Yi. Witness to non-termination of linear programs. Theoretical Computer Science. 681 (2017) 75-100.

[8] Liu J, Xu M, Zhan NJ, Zhao HJ. Discovering non-terminating inputs for multi-path polynomial programs. J. Syst. Sci. Complex. 27, 2014, 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 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI2018). ACM, 2018, 135-150.

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

[11] Leike J, Heizmann M. Geometric Nontermination Arguments. In 24th International conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS2018). Springer, Berlin, Heidelberg. 2018, 10806, 266-283.

[12] Colón M, Sipma H. Synthesis of linear ranking functions. In: Margaria, T., Yi, W. (eds.) In Proceedings of the 7th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'01), Springer, 2001, 67-81

[13] Podelski A, Rybalchenko A. A complete method for the synthesis of linear ranking functions //Proceedings of the Verification, model checking, and abstract interpretation (VMCAI), Berlin: Springer, 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, 215, 2012, 47-67

[15] Bagnara R, Mesnard F. Eventual Linear Ranking Functions. Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming (PPDP), Madrid, Spain, 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. 2016 10th International Symposium on Theoretical Aspects of Software Engineering (TASE'16), IEEE, 2016, 30–37.

[17] Leike J and Heizmann M. Ranking templates for linear loops. Proceedings of the 20th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'14), Springer, Berlin, Heidelberg. 2014, 8413, 172-186.

[18] Amir M. Ben-Amram, Samir Genaim. On Multiphase-Linear Ranking Functions. Computer Aided Verification. Springer, Cham. 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. International Colloquium on Theoretical Aspects of Computing (ICTAC 2007), Berlin:Springer, 2007, 34-49.

[20] Cousot P. Proving program invariance and termination by parametric abstraction, Langrangian Relaxation and semidefinite programming, In Proceedings of the 6th international conference on verification, model checking, and abstract interpretation (VMCAI'05), Springer, 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. 26(2):291-301

[22] Li Y. Termination of semi-algebraic loop programs. International Symposium on Dependable software Engineering: Theories, Tools and Applications (SETTA'17), Spiringer, 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.

[24] C.-W. Hsu, C.-C. Chang, C.-J. Lin. A practical guide to support vector classification. Technical Report, Department of Computer Science, National Taiwan University. 2003

[25] 周志华.机器学习.清华大学出版社.
[26] The RegularChains Library. http://regularchains.org/index.html
[27] Redlog — Computing with Logic. http://redlog.eu/

附中文参考文献:

[1] 周志华.机器学习.清华大学出版社.