软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(7):2981−3001 [doi: 10.13328/j.cnki.jos.006857]

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

安全的混成系统神经网络控制器生成与验证*

赵庆晔, 王 豫, 李宣东

(计算机软件新技术国家重点实验室(南京大学), 江苏 南京 210023)

通信作者: 王豫, E-mail: yuwang_cs@nju.edu.cn

摘 要: 控制器生成是混成系统控制中的重要问题. 生成具有安全保证的控制器, 关系着混成系统在安全攸关领域的使用. 提出了一种为混成系统生成具有安全保证的神经网络控制器的方法. 神经网络控制器的安全性由与其同时生成的障碍证书保证. 为了生成安全的神经网络控制器, 首先确定控制器的网络结构, 并基于混成系统构造训练数据集; 然后, 根据保证控制器安全的障碍证书条件编码神经网络训练时的损失函数. 当训练完成后, 学习到的神经网络控制器对于训练数据集中的数据是安全的, 但对于整个混成系统可能并不安全. 为了检验学习到的控制器在整个系统上的安全性, 将其安全验证问题转化为一组混合整数规划问题, 并使用数值优化器求解, 以得到形式化保证的结果. 工作实现了安全神经网络控制器生成工具 SafeNC, 并评估了它在 8 个基准系统上的性能. 实验结果表明: SafeNC 可以生成包含 6 个隐藏层的具有 1 804 个神经元的安全神经网络控制器; 同时, 与现有方法相比, SafeNC 可为更复杂的系统生成安全的神经网络控制器, 更有效且更具扩展性.

关键词: 混成系统; 安全控制; 神经网络控制器; 障碍证书; 混合整数规划

中图法分类号: TP311

中文引用格式: 赵庆晔, 王豫, 李宣东. 安全的混成系统神经网络控制器生成与验证. 软件学报, 2023, 34(7): 2981–3001. http://www.jos.org.cn/1000-9825/6857.htm

英文引用格式: Zhao QY, Wang Y, Li XD. Safe Neural Network Controller Synthesis and Verification for Hybrid Systems. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 2981−3001 (in Chinese). http://www.jos.org.cn/1000-9825/6857.htm

Safe Neural Network Controller Synthesis and Verification for Hybrid Systems

ZHAO Qing-Ye, WANG Yu, LI Xuan-Dong

(State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

Abstract: Controller synthesis is a fundamental problem in hybrid system control. The synthesis of safe controllers is related to the use of hybrid systems in safety-critical fields. This study proposes a novel approach to synthesizing neural network controllers with safety guarantees for hybrid systems. The safety of neural network controllers is guaranteed by barrier certificates, which are simultaneously synthesized with the controllers. To learn safe neural network controllers, first, the network structures of the controllers are determined, and the training datasets are constructed based on the hybrid system. Then, the loss function of network training is encoded based on the barrier certificate conditions guaranteeing the safety of the controllers. When the training process completes, the learned controllers are safe on training datasets but may not be safe on the whole hybrid system. To verify the safety of the learned controllers on the whole system, this study transforms the certification of safety conditions into a group of mixed-integer programming problems and adopts the numerical optimization solver to get formally guaranteed results. The safe neural network controller synthesis tool SafeNC is implemented and its performance on 8 benchmark systems is evaluated. SafeNC successfully synthesizes large controllers with up to 6 hidden layers and 1 804 neurons. The experimental results show that SafeNC can deal with more complex systems, and is more effective and scalable than the existing methods.

Key words: hybrid system; safety control; neural network controller; barrier certificate; mixed-integer programming

* 基金项目: 国家自然科学基金(62172211, 62172210, 62172200); 江苏省自然科学基金(BK20202001)
本文由“形式化方法与应用”专题特约编辑董云卫教授、刘关俊教授、毛晓光教授推荐.
收稿时间: 2022-09-04; 修改时间: 2022-10-08; 采用时间: 2022-12-05; jos 在线出版时间: 2022-12-30

控制器生成是混成系统控制领域的重要研究问题. 如自动驾驶、航空航天、物联网以及机器人控制等都需要合适的控制器来控制系统正常运行[1−3]. 随着机器学习的发展以及对于神经网络的研究, 神经网络已在控制器生成领域发挥了一定的作用[4−6]. 研究表明, 只包含单个隐藏层的神经网络具有以任意精度拟合任何复杂函数的能力, 因而其可以作为生成控制器的理想模板与架构[7−9]. 然而, 现有报告表明, 被神经网络控制的系统可能会表现出不安全行为, 从而导致不可预测的人力物力的损失. 这极大地阻碍了神经网络控制器在安全攸关领域的使用[10−12]. 因此, 对于被神经网络控制器控制的混成系统, 最关键和最具挑战性的问题之一是如何生成安全的神经网络控制器, 以保证被控系统不会到达不安全状态或者表现出危险的行为[13−15].

现有一些工作基于机器学习技术生成安全的神经网络控制器, 以应对可能的不安全行为[16−18]. 一类工作考虑危险行为的发生概率, 将度量后的风险编码进损失函数中, 通过优化损失函数的期望以保证安全[19−21]. 在生成神经网络控制器时, 这些方法通过最小化危险行为的期望来降低它们发生的概率. 然而, 由于控制器的安全性是通过损失函数的期望保证的, 这些方法只能减少不安全行为, 而不能完全消除它们[22−24].

为了完全避免不安全行为, 另一类工作通过额外的安全保证技术, 如控制障碍证书、控制李雅普诺夫函数和安全的备用控制器, 来保证所生成的神经网络控制器的安全性[25−27]. 控制障碍证书和控制李雅普诺夫函数将神经网络控制器所控制的系统轨迹限制在给定的安全区域内, 以避免违反安全性[28,29]. 对于使用安全备用控制器的方法, 当使用当前控制器控制的系统可能发生违反安全性的行为时, 它们会将当前控制器切换到安全的备用控制器, 以避免潜在的不安全行为[30]. 然而, 这些方法的局限性在于控制障碍证书、控制李雅普诺夫函数和安全的备用控制器通常需要人为预先提供, 或者需要一些关于系统和控制器的先验知识[31−33].

为了在不依赖先验知识的情况下保证被控系统的安全性, 本文提出一种基于反馈驱动的方法来学习安全的神经网络控制器. 控制器的安全性由与其同时生成的障碍证书保证. 障碍证书是关于系统状态的连续可微函数, 它将系统轨迹可达集与不安全区域分开, 从而避免系统轨迹进入不安全区域[34]. 因此, 障碍证书的存在, 是系统安全性的充分条件. 本文方法的关键想法是: 基于障碍证书提供的安全保证来指导神经网络控制器的训练, 同时, 通过形式化验证来确保生成的控制器在整个系统上的安全性. 如果验证失败, 验证中的反例则反馈到控制器训练中, 在下一次迭代训练中增强控制器的安全性. 与上述方法相比, 本文的障碍证书为控制器提供了形式化的安全保证; 同时, 由于障碍证书是与控制器一起学习的, 因此本文方法不需要任何先验知识.

本文方法由两个交互部分组成: 学习模块和验证模块, 如图 1 所示. 学习模块同时训练神经网络控制器与障碍证书. 障碍证书同样使用神经网络表示. 学习模块确定所采用的神经网络结构, 基于系统构建训练数据集, 然后编码损失函数以训练控制器, 并保证其安全性. 当学习结束时, 学习模块得到一组候选的神经网络控制器与其对应的障碍证书. 由于候选控制器是基于训练数据集学习的, 所以其在整个系统状态空间上不一定安全, 需要对其安全性进行验证. 验证模块将候选控制器的安全验证问题转化为一组混合整数规划问题, 并使用数值优化器进行求解来验证原始问题. 如果验证成功, 那么候选控制器在整个系统上是安全的; 否则, 验证模块将验证失败的反例添加到学习模块的训练数据集中, 用于在下一次迭代中继续训练控制器.

图 1 反馈驱动方法框架

总而言之, 本文的主要贡献如下:

- 基于障碍证书提供的安全保证与验证反例提供的安全指导, 提出了一种为混成系统生成安全神经网络控制器的反馈驱动方法. 安全控制器生成是由学习模块和验证模块组成的迭代过程.
- 将神经网络学习、安全问题转化以及反例驱动技术相结合, 从而能够生成具有各种结构的安全的神经网络控制器.

- 实现了安全神经网络控制器生成工具 SafeNC, 并在 8 个基准示例上评估其性能. SafeNC 在所有用例上的平均运行时间为 244.38 s. 实验结果表明, SafeNC 比现有方法更有效且更具扩展性.

本文第 1 节介绍相关知识. 第 2 节展示学习模块如何学习安全的神经网络控制器. 第 3 节介绍验证模块如何验证所学习控制器的安全性. 第 4 节介绍整体算法与工具实现, 并进行实验评估. 第 5 节讨论相关工作. 第 6 节总结全文.

1 预备知识

本节简要介绍本文使用的相关概念和基本知识, 包括前馈神经网络、混成系统、系统安全性的定义, 最后介绍障碍证书.

1.1 前馈神经网络

本文关注使用前馈全连接神经网络作为模板生成的安全控制器. 一个 n+1 层(从第 0 层到第 n 层)的前馈全连接神经网络$\mathcal{N}$由元组$\langle \mathcal{X},w,b,\phi\rangle$组成, 每个元素定义如下.

- $\mathcal{X}=\{x_0,\ldots,x_n\}$表示神经网络每层神经元的输出值.
- $w=\{w_1,\ldots,w_n\}$表示与每层神经元关联的权重矩阵.
- $b=\{b_1,\ldots,b_n\}$表示与每层神经元关联的偏置向量.
- ϕ表示激活函数.

对于前馈全连接神经网络(下述简称为神经网络), 非输入层神经元的值 x_k 由前一层的值 x_{k-1}、权重矩阵 w_k 和偏置向量 b_k 计算. 令 z_k, k=1,…,n−1 表示在应用激活函数ϕ前第 k 层神经元的值, 神经网络$\mathcal{N}$的前向传播定义如下:

$$\begin{cases} z_k = w_k x_{k-1} + b_k, & k = 1,\ldots,n-1 \\ x_k = \phi(z_k), & k = 1,\ldots,n-1 \\ x_n = w_n x_{n-1} + b_n \end{cases} \tag{1}$$

由公式(1)的前向传播定义, 神经网络$\mathcal{N}$的后向传播定义如下:

$$\begin{cases} \dfrac{\partial x_n}{\partial x_{n-1}} = w_n \\ \dfrac{\partial x_k}{\partial z_k} = \phi'(z_k), & k = 1,\ldots,n-1 \\ \dfrac{\partial z_k}{\partial x_{k-1}} = w_k, & k = 1,\ldots,n-1 \end{cases} \tag{2}$$

神经网络$\mathcal{N}$接收外部输入作为输入层神经元值 x_0, 其输出$\mathcal{N}(x_0)$是输出层神经元的值 x_n, 即$\mathcal{N}(x_0)=x_n$.

1.2 混成系统

一个混成系统由多个连续动态系统组成, 首先介绍连续动态系统的定义.

定义 1(连续动态系统). 一个连续动态系统$\mathcal{S}$由一个动力学模型(plant)以及一个控制器(controller)组成, 定义为元组$\langle x,\Psi,I,f\rangle$, 其中, 每个元素定义如下:

- $x=\langle x_1,\ldots,x_n\rangle$表示系统状态变量向量, 其中, x_i 表示第 i 个系统变量.
- Ψ表示系统空间.
- $I\subseteq\Psi$表示系统的初始区域.
- f 表示局部利普希茨连续向量场, 定义系统演化方程.

连续动态系统$\mathcal{S}$的演化定义如下:

$$\dot{x} = f(x,u) \tag{3}$$

其中, u 是神经网络控制器$\mathcal{N}$产生的控制输出. 控制器$\mathcal{N}$接收实时的系统状态 x 并输出控制信号 $u=\mathcal{N}(x)$, 以控

制系统$\mathcal{S}$. 图 2 展示了一个由四层神经网络控制的连续动态系统.

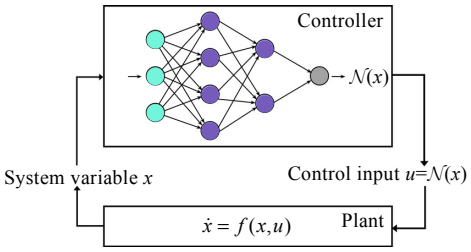

图 2 连续动态系统

定义 2(混成系统). 一个混成系统$\mathcal{S}$由多个连续动态系统组成, 定义为元组$\langle M,x,\Psi,I,T,R,f\rangle$, 其中, 每个元素定义如下.

- $M=\{M_1,\ldots,M_l\}$表示系统模态集合, 每个模态对应一个连续动态系统.
- $x=\langle x_1,\ldots,x_n\rangle$表示系统状态变量向量.
- $\Psi=\{\Psi_1,\ldots,\Psi_l\}$表示系统空间集合.
- $I=\{I_1,\ldots,I_l\}$表示初始区域集合.
- $M=\{t_{m,m'}|m\neq m'\}$, $T\subseteq\Psi$表示模态转换区域集合.
- $R=\{r_{m,m'}|m\neq m'\}$表示应用于模态转换时的重置函数集合.
- $f=\{f_1,\ldots,f_l\}$表示局部利普希茨连续向量场集合.

对于混成系统$\mathcal{S}$, 其控制器由一组神经网络构成: $\mathcal{N}=\{\mathcal{N}_1,\ldots,\mathcal{N}_l\}$, $\mathcal{N}_i=\langle\mathcal{X},w,b,\phi\rangle$. 在模态 M_i 下, 其控制输出 u_i 由控制器$\mathcal{N}_i$产生. 系统演化定义如下:

$$\dot{x}=f_i(x,\mathcal{N}_i(x)) \tag{4}$$

本文假设所关注的混成系统相关属性是已知的, 即混成系统的状态空间、初始区域、转换区域、重置函数以及系统动力学模型都是已知且具有明确定义. 动力学模型未知或相关属性未定义的混成系统不在本文工作考虑范围内.

1.3 混成系统安全性

混成系统的安全性是基于系统轨迹定义的, 首先介绍系统轨迹.

定义 3(轨迹). 对于混成系统$\mathcal{S}$, 如果所有模态 M_i 上的向量场 f_i 和控制器$\mathcal{N}_i$确定, 对于初始状态 $x_0\in I$, 系统的演化轨迹是唯一确定的. 给定初始时间 $t_0=0$、初始状态 $x_0\in I$ 和最大演化时间 $t\geqslant 0$, 系统$\mathcal{S}$的轨迹$\tau(t,x_0)$是一组关于系统模态和变量状态的序列:

$$(m_0,x_0),(m_1,x_1),\ldots,(m_i,x_i),(m_{i+1},x_{i+1}),\ldots \tag{5}$$

满足以下 3 个约束条件.

(1) 初始化条件: 在初始 $t_0=0$ 时刻, $(m_0,x_0)\in M\times I$.

(2) 连续演化连贯性条件: 如果 $m_i=m_{i+1}=m$, 则 x_i 到 x_{i+1} 的演化基于 f_m, 即 $x_{i+1}=f_m(x_i,\mathcal{N}_m(x_i))$.

(3) 离散演化连贯性条件: 如果 $x_i\in t_{m_i,m_{i+1}}$, 则 x_i 到 x_{i+1} 的演化基于重置函数 $r_{m_i,m_{i+1}}$, 即 $x_{i+1}=r_{m_i,m_{i+1}}(x_i)$.

定义 4(安全性). 对于混成系统$\mathcal{S}$以及给定的不安全区域 $U=\{U_1,\ldots,U_l\}$, 其中, $U\subseteq\Psi$且不与初始状态相交, 如果系统轨迹在任意时间都不会进入不安全区域, 则系统安全性得到保证. 定义如下:

$$\forall t\geqslant 0,\ \forall x\in I,\ \tau(t,x)\notin U \tag{6}$$

1.4 障碍证书

本文采用障碍证书来保证神经网络控制器及其所控系统的安全性. 在介绍障碍证书之前, 首先介绍李导数. 令$\mathcal{R}$表示实数, 对于连续可微函数 $k:\mathcal{R}^n\to\mathcal{R}$, 其对于自变量向量 $x=\langle x_1,\ldots,x_n\rangle$的导数定义如下:

$$\nabla k=\frac{\partial k}{\partial x}=\left(\frac{\partial k}{\partial x_1},\ldots,\frac{\partial k}{\partial x_n}\right) \tag{7}$$

定义 5(李导数). 对于向量场 $f(x,u)$, 连续可微函数 $k(x)$对于 $f(x,u)$的李导数$\mathcal{L}_{f(x,u)}k(x)$是∇k 与 $f(x,u)$对应元素的内积, 定义如下:

$$\mathcal{L}_{f(x,u)}k(x)=(\nabla k)\cdot f(x,u)=\frac{\partial k(x)}{\partial x}\cdot f(x,u)=\sum_{i=1}^{n}\frac{\partial k}{\partial x_i}(x)\cdot f^i(x,u) \tag{8}$$

其中, x_i 与 $f^i(x,u)$分别代表 x 与 $f(x,u)$中的第 i 个元素.

障碍证书是关于系统状态的连续可微函数, 其用具有相反符号的实数值映射系统轨迹与不安全状态. 障碍证书的零级集合作为一道屏障, 防止系统轨迹穿过它进入不安全状态. 混成系统的障碍证书的存在, 为系统安全性提供了保证[34].

定义 6(障碍证书条件). 对于混成系统$\mathcal{S}$的任意模态 $m\in M$ 以及对应转换区域 $t_{m,m'}$和重置函数 $r_{m,m'}$的模态对 (m,m'), 给定不安全区域 U, 其障碍证书$\mathcal{B}=\{\mathcal{B}_1,\ldots,\mathcal{B}_l\}$满足下述条件:

$$\begin{cases}\mathcal{B}_m(x)>0, & \forall x\in U_m\\ \mathcal{B}_m(x)\leqslant 0, & \forall x\in I_m\\ \mathcal{L}_{f_m(x,\mathcal{N}_m(x))}\mathcal{B}_m(x)\leqslant 0, & \forall \mathcal{B}_m(x)=0\\ \mathcal{B}_{m'}(x')\leqslant 0, & \forall x\in t_{m,m'},\ \mathcal{B}_m(x)\leqslant 0,\ x'=r_{m,m'}(x)\end{cases} \tag{9}$$

根据定义 6, 第 1 个条件要求障碍证书$\mathcal{B}$在不安全区域 U 上的输出值大于 0, 第 2 个条件要求$\mathcal{B}$在初始区域 I 上输出值小于等于 0. 因此, 障碍证书的零级集合$\mathcal{B}(x)=0$ 将不安全区域与初始区域分隔开. 第 3 个条件要求系统状态在连续演化过程中不会跨过$\mathcal{B}(x)=0$ 由小于等于 0 的状态变为大于 0 的状态. 第 4 个条件要求系统状态在两种模态之间转换时不会从由小于等于 0 的状态跃迁到大于 0 的状态. 如果混成系统$\mathcal{S}$的所有模态 M_i 都有对应满足定义 6 中条件的障碍证书$\mathcal{B}_i$, 那么系统的安全性得到保证.

2 安全神经网络控制器学习

对于混成系统$\mathcal{S}=\langle M,x,\Psi,I,T,R,f\rangle$, 目标是生成一组安全的神经网络控制器$\mathcal{N}=\{\mathcal{N}_1,\ldots,\mathcal{N}_l\}$控制系统轨迹不进入不安全区域. 本文使用障碍证书为控制器$\mathcal{N}$提供安全保证. 对于每个模态上的控制器$\mathcal{N}_i$, 为其生成对应的障碍证书$\mathcal{B}_i$. 障碍证书是与控制器一起学习的. 对于控制器$\mathcal{N}=\{\mathcal{N}_1,\ldots,\mathcal{N}_l\}$, 如果存在一组障碍证书$\mathcal{B}=\{\mathcal{B}_1,\ldots,\mathcal{B}_l\}$, 控制器与被控系统的安全性就得到了保证.

本文方法分为两个交互模块: 学习模块与验证模块. 学习模块基于训练数据集来训练神经网络控制器, 并使其尽可能满足安全性. 验证模块对所学到的候选控制器进行验证, 检验其对于整个混成系统是否依然满足安全性. 本节介绍学习模块的工作流程, 对于验证模块的介绍放在下一节. 首先定义神经网络结构; 然后, 基于混成系统构建数据集, 编码损失函数; 最后介绍安全控制器训练过程.

2.1 神经网络结构定义

在训练神经网络控制器$\mathcal{N}$之前, 其网络结构是预先定义的, 并且在训练过程中不改变. 假设$\mathcal{N}$为一组 $n_{\mathcal{N}}+1$ 层神经网络, 并在不同模态下有相同的结构和激活函数. 其输入层、隐藏层及输出层结构定义如下.

- 输入层: 包含的神经元数量与混成系统状态变量的数量相同. 控制器接收系统状态 x 作为其输入, 因此, 每个神经元对应一个系统状态变量. 系统状态变量的值即为输入层所接收的神经元值.
- 隐藏层: 可以包含任意多层数, 且每个隐藏层可以含有任意多个神经元. 激活函数应用于每个隐藏层神经元上. 本文支持多种激活函数, 如 ReLU、Sigmoid 和 tanh.
- 输出层: 控制器接收系统状态x并输出控制信号$u=\mathcal{N}(x)$以控制混成系统, 因此输出层的神经元个数与控制输入 u 的维度相同. 本文假设系统每个模态的控制输入是一维的. 所以控制器的输出层包含一个神经元. 对于控制输入为多个维度的系统, 更改输出层的神经元数量, 可以将控制输入扩展到多维.

为了生成安全的控制器$\mathcal{N}$, 学习模块同时学习一组障碍证书$\mathcal{B}=\{\mathcal{B}_1,\ldots,\mathcal{B}_l\}$为其提供安全保证. 障碍证书同样使用神经网络表示. 其接收系统状态 x 并输入值$\mathcal{B}(x)$. 与控制器类似, 假设$\mathcal{B}$为一组 $n_{\mathcal{B}}+1$层神经网络, 并在不同模态下具有相同的结构和激活函数. 其网络结构定义如下.

- 输入层: 包含与系统变量数量相同的神经元. 每个神经元对应一个系统状态变量.
- 隐藏层: 可以包含任意多层数与任意多神经元数. 支持多种激活函数.
- 输出层: 包含一个神经元. 神经元的值为障碍证书输出值$\mathcal{B}(x)$.

本文主要使用 ReLU 作为神经网络的激活函数. 本工作支持其他激活函数, 如 Sigmoid 和 tanh, 但 ReLU 在下节所述的验证过程中复杂度最低, 并且在实验中表现良好.

2.2 训练数据集构建

神经网络控制器以及障碍证书的训练是数据驱动过程, 其依赖对应的数据集来保证安全性. 对应于控制器$\mathcal{N}=\{\mathcal{N}_1,\ldots,\mathcal{N}_l\}$, 如果神经网络$\mathcal{B}=\{\mathcal{B}_1,\ldots,\mathcal{B}_l\}$是一组真正的障碍证书, 那么控制器的安全性得到保证. 为了训练控制器$\mathcal{N}$与障碍证书$\mathcal{B}$, 基于定义 6 的障碍证书条件构建相应的训练数据集.

为了使训练的神经网络满足第 1 个障碍证书条件"$\mathcal{B}_m(x)>0,\ \forall x\in U_m$", 学习模块从不安全区域 U 中采集数据点构建数据集 $D_1=\{x\in U_m\}$. 当基于 D_1 中的数据点训练神经网络时, 障碍证书的输出应为正值, 即$\mathcal{B}_m(x)>0$. 第 2 个障碍证书条件"$\mathcal{B}_m(x)\leqslant 0,\ \forall x\in I_m$"与第 1 个类似, 其对应数据集 $D_2=\{x\in I_m\}$包含从初始区域 I 中采集的数据点. 当基于 D_2 训练神经网络时, 障碍证书的输出应为非正值, 即$\mathcal{B}_m(x)\leqslant 0$.

为了满足第 3 个障碍证书条件"$\mathcal{L}_{f(x,\mathcal{N}_m(x))}\mathcal{B}_m(x)\leqslant 0,\ \forall \mathcal{B}_m(x)=0$", 第 3 个数据集 D_3 应该包含数据点$\{x|\mathcal{B}_m(x)=0\}$. 然而, 神经网络控制器以及障碍证书在训练过程中不断更新, 因此数据点$\{x|\mathcal{B}_m(x)=0\}$在不断变化. 为了构建合适的数据集, 学习模块在系统空间Ψ上采集数据点, 并在训练中实时进行测试, 以构建数据集 $D_3=\{x\in \Psi_m|\mathcal{B}_m(x)=0\}$. 此外, 由于浮点计算误差, 约束$\mathcal{B}_m(x)=0$ 很难被严格满足. 令$\eta_1>0$ 与$\eta_2>0$, 约束$\mathcal{B}_m(x)=0$被放缩为$-\eta_1\leqslant\mathcal{B}_m(x)\leqslant\eta_2$. 因此, 第 3 个条件的对应数据集为 $D_3=\{x\in \Psi_m|-\eta_1\leqslant\mathcal{B}_m(x)\leqslant\eta_2\}$. 当基于 D_3 中的数据点训练神经网络时, 其李导数应为非正值, 即 $\mathcal{L}_{f(x,\mathcal{N}_m(x))}\mathcal{B}_m(x)\leqslant 0$.

第 4 个数据集 D_4 对应于最后一个障碍证书条件"$\mathcal{B}_{m'}(x')\leqslant 0,\ \forall x\in t_{m,m'},\ \mathcal{B}_m(x)\leqslant 0,\ x'=r_{m,m'}(x)$". 学习模块从转移区域 $t_{m,m'}$中采集数据点, 并在运行时实时挑选满足条件的数据. 因此, D_4 被定义为$\{x\in t_{m,m'}|\mathcal{B}_m(x)\leqslant 0\}$.

2.3 损失函数编码

基于上一节构建的数据集, 所训练的神经网络控制器以及障碍证书在对应数据集上的输出条件如下:

$$\begin{cases}\mathcal{B}(x)>0, & \forall x\in D_1\\ \mathcal{B}(x)\leqslant 0, & \forall x\in D_2\\ \mathcal{L}_{f(x,\mathcal{N}(x))}\mathcal{B}(x)\leqslant 0, & \forall x\in D_3\\ \mathcal{B}(r_{m,m'}(x))\leqslant 0, & \forall x\in D_4\end{cases}\tag{10}$$

下面介绍如何编码对应于每个条件的损失函数.

对应于公式(10)中的第 1 个条件, 当基于数据集 D_1 训练时, 第 1 部分损失函数 l_1 旨在使障碍证书的输出$\mathcal{B}(x)$大于等于 0. 令$\mu>0$, l_1 编码如下:

$$l_1=\sum\nolimits_{x\in D_1}-\min(\mathcal{B}(x)-\mu,0)\tag{11}$$

对于 D_1 中的数据点 x, 如果$\mathcal{B}(x)\geqslant\mu$, 则 $l_1=0$; 否则, 最小化 l_1, 以训练神经网络满足条件$\mathcal{B}(x)\geqslant\mu$.

类似地, 第 2 部分损失函数 l_2 目标为使障碍证书在数据集 D_2 上输出值为小于等于 0, 编码如下:

$$l_2=\sum\nolimits_{x\in D_2}\max(\mathcal{B}(x),0)\tag{12}$$

对于 D_2 中的数据点 x, 如果$\mathcal{B}(x)\leqslant 0$, 则 $l_2=0$; 否则, 最小化 l_2, 使网络满足条件$\mathcal{B}(x)\leqslant 0$.

第 3 部分损失函数 l_3 旨在训练网络使 D_3 中的数据点对应的李导数 $\mathcal{L}_{f(x,\mathcal{N}(x))}\mathcal{B}(x)$ 小于等于 0, 保证系统轨迹在同一模态上连续演化时的安全性. 其编码如下:

$$l_3=\sum_{x\in D_3}\max(\mathcal{L}_{f(x,\mathcal{N}(x))}\mathcal{B}(x),0) \tag{13}$$

对于 D_3 中的数据点 x, 如果 $\mathcal{L}_{f(x,\mathcal{N}(x))}\mathcal{B}(x)\leqslant 0$, 则 l_3=0; 否则, 通过最小化 l_3 训练神经网络, 使其满足公式(10)中的第 3 个条件.

第 4 部分损失函数 l_4 旨在训练网络使 D_4 中的数据点在经过两个模态的跳转之后依旧对应于小于等于 0 的障碍证书输出值, 用以保证系统轨迹在不同模态之间离散演化时的安全性. 其编码如下:

$$l_4=\sum_{x\in D_4}\max(\mathcal{B}(r_{m,m'}(x)),0) \tag{14}$$

对于 D_4 中的数据点 x, 如果 $\mathcal{B}(r_{m,m'}(x))\leqslant 0$, 则 l_4=0; 否则, 最小化 l_4, 以训练神经网络满足系统在离散演化时的安全性条件.

最后, 神经网络训练的总体损失函数 l 编码为 4 个子损失函数的加权和. 令 $\alpha>0$, $\beta>0$, $\gamma>0$ 与 $\delta>0$ 表示 4 个子损失函数之间的权重, 总体损失函数 l 编码如下:

$$l=\alpha\cdot l_1+\beta\cdot l_2+\gamma\cdot l_3+\delta\cdot l_4 \tag{15}$$

2.4 安全控制器训练

学习模块通过最小化公式(15)中定义的损失函数 l, 以训练安全的神经网络控制器和与之对应的障碍证书. 学习模块首先确定神经网络的结构, 并基于混成系统构建训练数据集 D_1–D_4. 在训练中, 数据集被随机打乱并分为多个批次. 基于每个批次计算损失函数 l 并最小化 l, 以更新神经网络参数. 当所有批次对应的损失函数都为 0, 学习模块停止训练, 并将所学到的候选控制器传入下节所述的验证模块进行安全验证: 如果验证成功, 候选控制器在整个混成系统上是真正安全的控制器; 否则, 验证模块会返回不安全的反例到学习模块. 学习模块将这些反例加入对应的数据集中, 并在下一次迭代中继续训练控制器. 学习模块与验证模块的反馈迭代过程持续进行, 直到生成一组安全的神经网络控制器为止.

为了使安全控制器生成更容易成功, 本文基于训练数据的代表性、泛化性与验证复杂度以及验证模块返回的反例对学习模块进行改进.

- 增强训练数据的代表性

候选控制器是基于训练数据集训练的. 增强训练数据集的代表性, 可以使候选控制器更容易在整个混成系统上被验证成功. 一种直接的方法是, 增加训练数据集的大小以采样更多的数据点. 此外, 由于混成系统以及神经网络的连续性, 某个数据点与其周围的数据点的性质是类似的. 例如, 如果数据点 x_0 满足 $\mathcal{B}(x_0)>\sigma>0$, 那么在 x_0 周围一定存在一个 λ 邻域, 其中的数据点 x 满足下式:

$$\mathcal{B}(x)>0,\ \forall\|x-x_0\|_2<\lambda \tag{16}$$

对于数据集 D_1, 如果满足 $\mathcal{B}(x)>\sigma$ 的数据点 x 的 λ 邻域可以覆盖整个不安全区域 U, 那么学习模块训练出来的障碍证书则是关于 U 的真正的障碍证书. 因此, 为了提高训练数据点在其邻域上的代表性, 令 $\sigma_1>0$, $\sigma_2>0$, $\sigma_3>0$, $\sigma_4>0$, 定义于公式(11)–(14)的子损失函数改进编码如下:

$$\begin{cases} l_1=\sum_{x\in D_1}-\min(\mathcal{B}(x)-\mu,\sigma_1)\\ l_2=\sum_{x\in D_2}\max(\mathcal{B}(x),-\sigma_2)\\ l_3=\sum_{x\in D_3}\max(\mathcal{L}_{f(x,\mathcal{N}(x))}\mathcal{B}(x),-\sigma_3)\\ l_4=\sum_{x\in D_4}\max(\mathcal{B}(r_{m,m'}(x)),-\sigma_4) \end{cases} \tag{17}$$

- 增强泛化性与降低验证复杂度

观察公式(10)可以发现, 其中 4 个不等式条件的符号右侧都是 0. 也就是说, 每个条件都是与 0 做对比. 因此, 对于左侧来说, 只需关注其值的正负号, 而值的大小对于等式的成立没有影响. 基于此, 本文在神经网络训练中, 对障碍证书的参数增加正则化损失 l_{reg}. 正则化损失可以增强神经网络的泛化性, 并且可以降低神经网络参数的范数大小. 网络参数范数的收缩降低了神经元值的取值范围, 可以降低下节所述验证模块求解时的复杂度. 令 $\zeta>0$ 表示正则化损失 l_{reg} 的权重系数, 公式(15)定义的总体损失 l 改进如下:

$$l=\alpha\cdot l_1+\beta\cdot l_2+\gamma\cdot l_3+\delta\cdot l_4+\zeta\cdot l_{reg} \tag{18}$$

- 扩展反例数据点

当验证模块对候选控制器的安全验证失败时, 其将返回导致验证失败的反例. 由于混成系统以及神经网络具有连续性, 反例点周围的数据点可能拥有和反例点类似的性质, 因而也可能是反例. 因此, 学习模块在验证模块返回的反例周围采样更多的数据点添加到训练数据集中, 以增强反馈驱动的效果. 在下一轮迭代中, 基于这些数据点继续训练控制器, 使其在整个数据集上安全.

3 神经网络控制器安全性验证

当神经网络训练结束时, 学习模块得到一组候选的神经网络控制器$\mathcal{N}$与障碍证书$\mathcal{B}$.控制器的安全性由障碍证书保证. 在训练数据集上, $\mathcal{B}$是真正的障碍证书且$\mathcal{N}$是安全的控制器; 然而在整个系统上, 候选障碍证书$\mathcal{B}$可能无法保证控制器$\mathcal{N}$的安全性. 本节介绍验证模块如何对候选控制器进行安全验证.

如第 1.4 节所介绍, 如果所学的候选障碍证书在混成系统$\mathcal{S}$上满足 4 个障碍证书条件, 那么候选障碍证书是一个真正的障碍证书, 候选控制器在系统$\mathcal{S}$上的安全性得到保证. 因此, 候选控制器的安全验证问题可以通过验证候选障碍证书是否满足定义 6 中的障碍证书条件推断. 然而, 由于混成系统以及神经网络的非线性与复杂性, 直接对定义 6 中的障碍证书条件进行验证非常困难. 本文将对于 4 个障碍证书条件的验证问题转化为一组对应的混合整数规划问题, 然后调用数值优化器对转化后的问题进行求解与验证.

3.1 第1个障碍证书条件验证

首先考虑第 1 个障碍证书条件"$\mathcal{B}_m(x)>0, \forall x\in U_m$". 此条件要求障碍证书的输出$\mathcal{B}_m(x)$在不安全区域$U_m$上大于 0. 根据第 2.1 节的定义, 候选障碍证书$\mathcal{B}$为一组 $n_{\mathcal{B}}+1$层神经网络. 其输出层 $x_{n_{\mathcal{B}}}$ 包含一个神经元且 $x_{n_{\mathcal{B}}}=\mathcal{B}_m(x_0)$. 因此, 对于任意 $x_0\in U_m$, 如果其输出层神经元值 $x_{n_{\mathcal{B}}}$ 都大于 0, 那么第 1 个障碍证书条件被验证成功. 基于公式(1)中的神经网络前向计算, 第 1 个障碍证书条件验证问题被转化为关于在不安全区域上候选障碍证书输出层神经元最小值的优化问题, 定义如下:

$$\begin{cases}\underset{x_0}{\text{minimize}}\ x_{n_{\mathcal{B}}}\\ \text{s.t. } x_0\in U_m,\ \forall m\in M\\ \quad x_{n_{\mathcal{B}}}=w_{n_{\mathcal{B}}}x_{n_{\mathcal{B}}-1}+b_{n_{\mathcal{B}}}\\ \quad x_k=\phi(z_k),\ k=1,\ldots,n_{\mathcal{B}}-1\\ \quad z_k=w_kx_{k-1}+b_k,\ k=1,\ldots,n_{\mathcal{B}}-1\end{cases} \tag{19}$$

在问题(19)中:

- x_0表示候选障碍证书$\mathcal{B}_m$的输入, 为不安全区域 U 中系统状态的值.
- $x_{n_{\mathcal{B}}}$表示$\mathcal{B}_m$输出层神经元的值, 为候选障碍证书的输出.
- z_k与 x_k分别表示在应用激活函数前后候选障碍证书第 k 层神经元的值.

如果问题(19)的全局最小值大于 0, 那么候选障碍证书$\mathcal{B}_m$则满足第 1 个障碍证书条件.

本文主要关注的激活函数ϕ是 ReLU, 定义如下:

$$ReLU(x)=\begin{cases}x, & x>0\\ 0, & x\leqslant 0\end{cases} \tag{20}$$

可以发现, ReLU 激活函数是非线性的. 为消除非线性以加速问题(19)的优化求解过程, 本文将 $y=ReLU(x)$ 转化为一组等价的混合整数规划约束[35].

定理 1(ReLU 的混合整数规划编码). 令$[l_x,u_x]$表示 x 的输入域, y 表示 $ReLU(x)$的值域, $y=ReLU(x)$被等价编码为下述 5 个线性与二进制变量约束:

$$\begin{cases} -x+y-l\cdot t\leqslant -l \\ x-y\leqslant 0 \\ y-u\cdot t\leqslant 0 \\ y\geqslant 0 \\ t\in\{0,1\} \end{cases} \tag{21}$$

其中, t 代表二进制中间变量, 取值为 0 或 1; l 是 x 取值范围的下界估计, 满足 $l\leqslant l_x$; u 是 x 取值范围的上界估计, 满足 $u\geqslant u_x$.

根据定理 1, 如果 $x<0$, 则 $y=0$ 且 $t=0$; 如果 $x>0$, 则 $y=x$ 且 $t=1$; 如果 $x=0$, 则 $y=0$ 且 $t=0$ 或 $t=1$. 本文简要证明 $x<0$ 可以推论出 $y=0$ 且 $t=0$. 另外两个推论的证明类似并省略.

证明: 当 $x<0$ 时, 假设 $t=1$. 此时, 公式(21)的第 1 个约束为 $y\leqslant x$, 第 2 个约束为 $y\geqslant x$, 即 $y=x$. 又因为 $x<0$, 所以有 $y<0$, 与第 4 个约束 $y\geqslant 0$ 矛盾. 所以当 $x<0$ 时, $t\neq 1$. 对于 $t=0$, 可以发现, 5 个约束之间没有矛盾. 根据第 3 个约束 $y\leqslant 0$ 与第 4 个约束 $y\geqslant 0$ 可以推断出 $y=0$. 因此, 如果 $x<0$, 则 $y=0$ 且 $t=0$.

将问题(19)中的 ReLU 激活函数转化为定理 1 中的混合整数规划编码之后, 问题(19)转化为如下优化问题:

$$\begin{cases} \underset{x_0}{\text{minimize}}\ x_{n_{\mathcal{B}}} \\ \text{s.t. } x_0\in U_m,\ \forall m\in M \\ \quad x_{n_{\mathcal{B}}}=w_{n_{\mathcal{B}}}x_{n_{\mathcal{B}}-1}+b_{n_{\mathcal{B}}} \\ \quad -z_k+x_k-l_kt_k\leqslant -l_k,\ k=1,...,n_{\mathcal{B}}-1 \\ \quad z_k-x_k\leqslant 0,\ k=1,...,n_{\mathcal{B}}-1 \\ \quad x_k-u_kt_k\leqslant 0,\ k=1,...,n_{\mathcal{B}}-1 \\ \quad x_k\geqslant 0,\ k=1,...,n_{\mathcal{B}}-1 \\ \quad t_k\in\{0,1\}^{d_k},\ k=1,...,n_{\mathcal{B}}-1 \\ \quad z_k=w_kx_{k-1}+b_k,\ k=1,...,n_{\mathcal{B}}-1 \end{cases} \tag{22}$$

其中, t_k 表示二进制变量向量, d_k 表示 x_k 的维数, 神经元取值的上下界估计 u_k 与 l_k 通过区间计算逐层推断[36].

对于问题(22), 验证模块使用数值优化器 Gurobi 9.5[37]寻找全局的问题最优解. 对于约束 $x_0\in U_m$ 中可能存在的非线性, 优化器提供了一种内置的分段线性逼近方法, 可以将其转化为一组分段线性函数, 构成相应的一组混合整数线性规划约束. 然而, 当求解问题(22)的全局最优解时, 由于存在浮点计算误差, 优化器很难找到精确的全局最优解. 下一节介绍如何基于优化器返回的当前解对所学习的候选障碍证书 $\mathcal{B}_m$ 是否满足第 1 个障碍证书条件进行形式化验证.

3.2 基于当前解的障碍证书条件验证

当优化器对问题(22)进行求解时, 它会实时返回优化器找到的当前解 p、与当前解 p 和全局最优解 p^*之间的最大相对误差ξ. 其中, ξ满足如下条件:

$$\xi\geqslant\frac{|p-p^*|}{|p|} \tag{23}$$

根据公式(23), 如果$\xi<1$ 且 $p\neq 0$, 那么有 $pp^*>0$. 即可以根据当前解 p 的符号与ξ的值推断全局最优解 p^*的符号. 证明如下:

$$\left.\begin{aligned} &\xi\geqslant\frac{|p-p^*|}{|p|}\wedge\xi<1\wedge p\neq 0\Rightarrow\frac{|p-p^*|}{|p|}<1\Rightarrow|p-p^*|^2<|p|^2\Rightarrow \\ &p^2-2pp^*+(p^*)^2<p^2\Rightarrow pp^*>\frac{1}{2}(p^*)^2\Rightarrow pp^*>0 \end{aligned}\right\} \tag{24}$$

推论 1. 令 p 为问题(22)的当前解, p^*为全局最优解, ξ为当前解 p 与全局最优解 p^*之间的最大相对误差, 有如下推断:

$$p>0\wedge\xi<1\Rightarrow p^*>0.$$

证明: 当优化器对问题(22)进行求解时, 若其返回的当前解 $p>0$ 且最大误差 $\xi<1$, 根据公式(24), 可以推断出全局最优解 $p^*>0$, 说明所有不安全区域上的障碍证书输出值都大于 0. 所学习的候选障碍证书 $\mathcal{B}_m$ 满足第 1 个障碍证书条件"$\mathcal{B}_m(x)>0, \forall x\in U_m$". 第 1 个障碍证书条件得到验证.

3.3 第2个障碍证书条件验证

第 2 个障碍证书条件"$\mathcal{B}_m(x)\leqslant 0, \forall x\in I_m$"的验证与第 1 个是同构的, 除了 x_0 的验证区域以及问题优化方向不同. 此条件要求在初始区域 I 上候选障碍证书 $\mathcal{B}_m$ 的输出应小于等于 0. 本文将对其的验证问题转化为在初始区域上求解障碍证书输出最大值的优化问题, 定义如下:

$$\begin{cases}\underset{x_0}{\text{maximize}}\ x_{n_{\mathcal{B}}}\\ \text{s.t. } x_0\in I_m,\ \forall m\in M\\ \quad x_{n_{\mathcal{B}}}=w_{n_{\mathcal{B}}}x_{n_{\mathcal{B}}-1}+b_{n_{\mathcal{B}}}\\ \quad x_k=\phi(z_k),\ k=1,...,n_{\mathcal{B}}-1\\ \quad z_k=w_kx_{k-1}+b_k,\ k=1,...,n_{\mathcal{B}}-1\end{cases}\tag{25}$$

由于问题(25)与问题(19)的同构性, 问题(19)所采用的处理方法可以应用于问题(25). 本文省略转换过程以及转换后的问题描述.

推论 2. 令 p 为问题(25)的当前解, p^* 为全局最优解, ξ 为当前解 p 与全局最优解 p^* 之间的最大相对误差, 有如下推断:

$$p<0\wedge\xi<1\Rightarrow p^*<0.$$

证明: 推论 2 的证明与推论 1 类似. 当优化器对问题(25)进行求解时, 若 $p<0$ 且 $\xi<1$, 那么全局最优解 $p^*<0$, 说明所有初始区域上的障碍证书输出值都小于等于 0. 因此, 第 2 个障碍证书条件"$\mathcal{B}_m(x)\leqslant 0, \forall x\in I_m$"得证.

3.4 第3个障碍证书条件验证

第 3 个障碍证书条件"$\mathcal{L}_{f(x,\mathcal{N}_m(x))}\mathcal{B}_m(x)\leqslant 0, \forall \mathcal{B}_m(x)=0$"保证系统在同一模态连续演化时的安全性. 系统轨迹一旦到达障碍证书的零级集合, 即 $\mathcal{B}_m(x)=0$, 系统将沿着此集合演化或者从它反弹. 此条件要求在候选栅栏函数的零级集合内的系统状态的李导数要小于等于 0. 其验证问题转化为如下优化问题:

$$\begin{cases}\underset{x}{\text{maximize}}\ s\\ \text{s.t. } x\in\Psi_m,\ \forall m\in M\\ \quad s=\mathcal{L}_{f_m(x,\mathcal{N}_m(x))}\mathcal{B}_m(x)\\ \quad \mathcal{B}_m(x)=0\end{cases}\tag{26}$$

其中, s 是表示李导数的中间变量, $\mathcal{N}_m(x)$ 和 $\mathcal{B}_m(x)$ 分别是在模态 m 上候选控制器和障碍证书的输出. 如果问题(26)的全局最大值小于等于 0, 那么第 3 个障碍证书条件被验证满足.

根据公式(8), 将李导数转化为内积形式, 并将两个神经网络 $\mathcal{N}_m(x)$ 和 $\mathcal{B}_m(x)$ 的前向计算展开, 问题(26)进一步转化为如下形式:

$$\begin{cases}\underset{x_0}{\text{maximize}}\ s\\ \text{s.t. } x_0\in\Psi_m,\ \forall m\in M\\ \quad s=\dfrac{\partial\mathcal{B}_m(x_0)}{\partial x_0}\cdot f_m(x_0,\mathcal{N}_m(x_0))\\ \quad w_{n_{\mathcal{B}}}x_{n_{\mathcal{B}}-1}+b_{n_{\mathcal{B}}}=0\\ \quad x_k=\phi(w_kx_{k-1}+b_k),\ k=1,...,n_{\mathcal{B}}-1\\ \quad \mathcal{N}_m(x_0)=p_{n_{\mathcal{N}}}y_{n_{\mathcal{N}}-1}+q_{n_{\mathcal{N}}}\\ \quad y_k=\psi(p_ky_{k-1}+q_k),\ k=2,...,n_{\mathcal{N}}-1\\ \quad y_1=\psi(p_1x_0+q_1)\end{cases}\tag{27}$$

在问题(27)中: 候选控制器$\mathcal{N}_m$为一个$n_{\mathcal{N}}+1$层神经网络, 如第 2.1 节中所定义; p_k和 q_k分别是控制器$\mathcal{N}_m$的权重矩阵和偏置向量; y_k表示控制器$\mathcal{N}_m$第 k 层的输出; w_k和 b_k分别是候选障碍证书$\mathcal{B}_m$的权重矩阵和偏置向量; ψ和ϕ分别是控制器$\mathcal{N}_m$和障碍证书$\mathcal{B}_m$的激活函数.

现在考虑 $\dfrac{\partial \mathcal{B}_m(x_0)}{\partial x_0}$ 的编码. 根据公式(2)中神经网络的后向计算以及 $x_{n_B}=\mathcal{B}_m(x_0)$, $\dfrac{\partial \mathcal{B}_m(x_0)}{\partial x_0}$ 编码如下:

$$\frac{\partial \mathcal{B}_m(x_0)}{\partial x_0}=\frac{\partial x_{n_B}}{\partial x_0}=w_{n_B}\cdot\phi'(z_{n_B-1})\times w_{n_B-1}\ldots w_2\cdot\phi'(z_1)\times w_1 \tag{28}$$

本文主要使用 ReLU 作为障碍证书的激活函数. 根据公式(20)可以发现, $ReLU(x)$除了 x=0 之外是可导的. 当x=0时, 其左导数 $\lim\limits_{x\to 0^-} ReLU'(x)=0$, 而右导数 $\lim\limits_{x\to 0^+} ReLU'(x)=1$. 本文采用机器学习中常见的导数定义方法, $ReLU'(x)$定义如下[38]:

$$ReLU'(x)=\begin{cases}1, & x>0\\ 0, & x\leqslant 0\end{cases} \tag{29}$$

基于上式, $ReLU'(x)$在 x=0 处可导. 可以在整个系统空间上计算 $\dfrac{\partial \mathcal{B}_m(x_0)}{\partial x_0}$.

回想定理 1 中对 ReLU 激活函数的混合整数规划编码, 有 x<0 时 t=0; x>0 时 t=1; x=0 时 t=0 或 t=1. 与公式(29)相比, 可以发现, 当 $x\neq 0$ 时, $t=ReLU'(x)$. 本文使用二进制变量 t 作为 $ReLU(x)$的导数.

当 x=0 时, $ReLU'(x)$=0 而 t=0 或 1. 使用 t 替代 $ReLU'(x)$是原值的过近似. 在问题(27)优化过程中, 求解器会遍历 t=0 与 t=1 两个分支, 然后返回当前最优值. 如果过近似后的最优值满足验证所期望的条件, 那么原始最优值同样也满足此条件.

基于上述讨论, $\dfrac{\partial \mathcal{B}_m(x_0)}{\partial x_0}$ 的计算进一步编码如下:

$$\frac{\partial \mathcal{B}_m(x_0)}{\partial x_0}=w_{n_B}\cdot t_{n_B-1}\times w_{n_B-1}\ldots w_2\cdot t_1\times w_1 \tag{30}$$

其中, t_k表示候选障碍证书第 k 层神经元中使用混合整数规划对 ReLU 编码引入的二进制变量向量.

在求解问题(27)时, ReLU 激活函数根据定理 1 转化为一组混合整数规划约束. 对于 $x_0\in\Psi_m$中可能存在的非线性以及神经网络$\mathcal{N}_m$和$\mathcal{B}_m$可能包含的非线性激活函数, 优化器使用内置的分段线性逼近方法将其转化为一组对应的混合整数线性规划约束.

推论 3. 令 p 为问题(27)的当前解, p^*为全局最优解, ξ为当前解 p 与全局最优解 p^*之间的最大相对误差, 有如下推断:

$$p<0\wedge\xi<1\Rightarrow p^*<0.$$

证明: 当优化器对问题(27)进行求解时, 若 p<0 且ξ<1, 则全局最优解 p^*<0, 说明所有在候选障碍证书的零级集合内的系统状态的李导数都小于等于 0. 故, 第 3 个障碍证书条件“$\mathcal{L}_{f(x,\mathcal{N}_m(x))}\mathcal{B}_m(x)\leqslant 0,\ \forall \mathcal{B}_m(x)=0$”得证.

3.5 第4个障碍证书条件验证

第 4 个障碍证书条件“$\mathcal{B}_{m'}(x')\leqslant 0,\ \forall x\in t_{m,m'},\ \mathcal{B}_m(x)\leqslant 0,\ x'=r_{m,m'}(x)$”保证系统在不同模态之间跳转时的安全性. 当系统轨迹状态 x 在模态 m 的跳转区域内且此时障碍证书输出值$\mathcal{B}_m(x)$小于等于 0, 那么其跳转之后在模态 m' 的状态 $x'=r_{m,m'}(x)$所对应的障碍证书输出值$\mathcal{B}_{m'}(x)$也应小于等于 0. 其验证问题转化为如下优化问题:

$$\begin{cases}\underset{x_0}{\text{maximize}}\ x'_{n_{\mathcal{B}}} \\ \text{s.t. } x_0 \in t_{m,m'},\ x'_0 = r_{m,m'}(x_0) \\ \quad z_k = w_k x_{k-1} + b_k,\ z'_k = w'_k x'_{k-1} + b_{k'},\ k = 1,\ldots,n_{\mathcal{B}} - 1 \\ \quad x_k = \phi(z_k),\ x'_k = \phi(z'_k),\ k = 1,\ldots,n_{\mathcal{B}} - 1 \\ \quad w_{n_{\mathcal{B}}} x_{n_{\mathcal{B}}-1} + b_{n_{\mathcal{B}}} \leqslant 0,\ x'_{n_{\mathcal{B}}} = w'_{n_{\mathcal{B}}} x'_{n_{\mathcal{B}}-1} + b'_{n_{\mathcal{B}}}\end{cases} \tag{31}$$

在问题(31)中, 约束的左侧部分为在模态 m 上障碍证书$\mathcal{B}_m$ 的计算, 右侧部分为在模态 m'上障碍证书$\mathcal{B}_{m'}$的计算. $x'_{n_{\mathcal{B}}}$ 表示模态 m'上障碍证书$\mathcal{B}_{m'}$的输出.

推论 4. 令 p 为问题(31)的当前解, p^*为全局最优解, ξ为当前解 p 与全局最优解 p^*之间的最大相对误差, 有如下推断:

$$p<0 \wedge \xi<1 \Rightarrow p^*<0.$$

证明: 当优化器对问题(31)进行求解时, 若 $p<0$ 且$\xi<1$, 那么全局最优解 $p^*<0$, 保证系统在不同模态之间跳转时的安全性. 第 4 个障碍证书条件“$\mathcal{B}_{m'}(x')\leqslant 0, \forall x\in t_{m,m'}, \mathcal{B}_m(x)\leqslant 0, x'=r_{m,m'}(x)$”得证.

3.6 安全控制器验证

在对问题(22)、(25)、(27)、(31)进行求解时, 如果相对误差ξ小于 1, 优化终止. 基于推论 1–4, 下述推论保证了候选神经网络控制器在整个混成系统上的安全性.

推论 5. 当求解问题(22)、(25)、(27)、(31)时, 令 p_1, p_2, p_3, p_4 为当前解, $p_1^*, p_2^*, p_3^*, p_4^*$ 为最优解, $\xi_1, \xi_2, \xi_3, \xi_4$ 为最大相对误差, 如果 $p_1>0 \wedge \xi_1<1 \wedge p_2<0 \wedge \xi_2<1 \wedge p_3<0 \wedge \xi_3<1 \wedge p_4<0 \wedge \xi_4<1$, 那么 $p_1^*>0 \wedge p_2^*<0 \wedge p_3^*<0 \wedge p_4^*<0$. 定义 6 中的 4 个障碍证书条件被证明是满足的. 候选控制器被验证是安全的.

推论 5 的证明可以由推论 1–4 得到. 另一方面, 如果优化器返回的当前解 p 带有不期望的正负号, 且此时的相对误差$\xi<1$, 例如求解问题(22)时 $p<0$ 且$\xi<1$, 那么此时当前解 p 所对应的优化参数 x_0 是一个反例(除了对于问题(31)以外, 因为在求解问题(31)时存在对于 $ReLU'(x)$的过近似). 此时, 候选控制器在整个混成系统上不是安全的. 验证模块将反例 x_0 返回到学习模块中, 用以在下一轮迭代中继续训练控制器.

4 实现与实验

本节首先展示本文所提出的反馈驱动方法的算法, 并介绍安全神经网络控制器生成工具 SafeNC 的实现. 我们在 2 个混成系统以及 6 个连续动态系统上对 SafeNC 进行评估, 并与现有方法进行对比.

4.1 算法与工具实现

算法 1 展示了本文提出的生成安全神经网络控制器的反馈驱动方法 SafeNC. 为了生成安全的控制器, SafeNC 同时生成一组障碍证书为控制器提供安全保证. SafeNC 由学习模块和验证模块迭代构成. 在反馈迭代前, 首先定义候选控制器和障碍证书的结构, 并对其进行初始化; 然后, 基于混成系统构建训练数据集. 学习模块基于训练数据集更新控制器以及障碍证书参数, 使其在数据集上满足安全性. 验证模块将安全验证问题转化为一组混合整数规划问题, 并进行求解验证: 如果验证成功, SafeNC 终止并返回安全的神经网络控制器; 否则, 验证中的反例被扩展并添加到训练数据集中进行下一次迭代训练. 若在到达最大迭代次数之前未生成安全的控制器, 则 SafeNC 返回失败.

算法 1. SafeNC.

Input: Hybrid system $\mathcal{S}$, maximum epoch E.

Output: Safe neural network controllers $\mathcal{N}$.

(1) Define the structures of neural network controllers $\mathcal{N}$ and barrier certificates $\mathcal{B}$

(2) Initialize $\mathcal{N}$ and $\mathcal{B}$ with parameters $\theta_{\mathcal{N}}$ and $\theta_{\mathcal{B}}$ and generate training datasets D_1–D_4

(3) **For** e=0,…,E **do**

(4) Construct loss function l according to Eq.(18)

(5) **While** $l>0$ **do**

(6) Update $\theta_{\mathcal{N}}$ and $\theta_{\mathcal{B}}$ by minimizing l

(7) Construct Problems (22), (25), (27), (31) for safety verification

(8) Optimize Problems (22), (25), (27), (31) and get optimums p_1, p_2, p_3, p_4 with errors ξ_1, ξ_2, ξ_3, ξ_4

(9) **If** $p_1>0\wedge\xi_1<1\wedge p_2<0\wedge\xi_2<1\wedge p_3<0\wedge\xi_3<1\wedge p_4<0\wedge\xi_4<1$ **then**

(10) **Return** $\mathcal{N}$

(11) **Else**

(12) Sample data points around counterexamples and add them to corresponding training datasets

(13) **Return** *Node*

SafeNC 算法是可靠的(sound)但不是完备的(not complete). 其可靠性指 SafeNC 返回的神经网络控制器一定是安全的. 本文第 3 节介绍了如何将候选控制器的安全验证问题转化为一组混合整数规划问题, 并讨论了如何根据当前解与最大相对误差推断全局最优解的符号, 因此, SafeNC 返回的神经网络控制器的安全性是得到形式化保证的. 不完备性指 SafeNC 不保证一定返回安全的控制器. 由于神经网络训练的随机性, SafeNC 算法不保证可以在最大迭代次数内生成安全的神经网络控制器.

SafeNC 的学习模块是基于 TensorFlow 2.0[39]实现的, 验证模块调用数值优化器 Gurobi 9.5[37]对候选控制器进行安全验证. 所有实验均运行在安装 Ubuntu 18.04 系统的服务器上, 配置为 320 GB 内存, 两个 3.20 GHz Intel Xeon Gold 6146 CPU 和 3 个 NVIDIA TITAN V GPU. SafeNC 使用的超参数默认设置为 $\eta_1=\eta_2=0.5$, $\mu=\sigma_1=\sigma_2=\sigma_3=\sigma_4=0.005$, $\alpha=\beta=\gamma=\delta=\zeta=1$. 其他超参数比如神经网络结构、训练迭代轮数、训练数据集大小等, 根据不同实验用例的特性进行启发式设置.

4.2 用例展示

例 1(ACAS Xu 连续动态系统[40]): 图 3 展示了飞行控制 ACAS Xu 系统的示意图. 两架飞机在空中的飞行轨迹可能交汇. 为了避免发生碰撞, 需要生成一个控制器以控制飞机 1 的飞行方向. 系统状态由两架飞机之间的相对距离$[x,y]$、两架飞机的飞行速度 v_1 与 v_2 以及飞机 1 的飞行方向 θ描述. 假定飞机 2 的飞行方向不变, 两架飞机的飞行速度不变, 且初速度为 0.1.

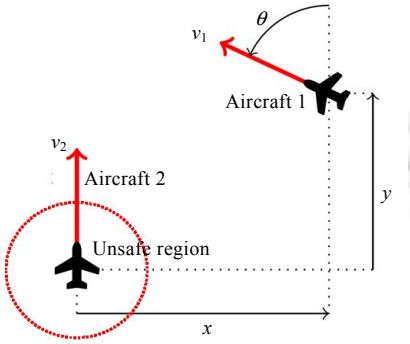

图 3 ACAS Xu 系统

系统的状态空间 Ψ定义如下:

$$\Psi=\{-1\leqslant x\leqslant 1,-1\leqslant y\leqslant 1,0\leqslant\theta\leqslant\pi,v_1=0.1,v_2=0.1\} \tag{32}$$

可能发生碰撞的不安全区域为围绕飞机 2 的圆形区域, 定义如下:

$$U=\{(x+1)^2+(y+1)^2\leqslant 0.25,0\leqslant\theta\leqslant\pi,v_1=0.1,v_2=0.1\}\cap\Psi \tag{33}$$

系统考虑的碰撞情况为: 飞机 1 从飞机 2 的右前方区域以任意方向朝向飞机 2 飞行. 系统初始状态定义如下:

$$I=\{x=1,0.5\leqslant y\leqslant 1,0\leqslant \theta\leqslant \pi,v_1=0.1,v_2=0.1\} \tag{34}$$

系统的演化方程定义如下:

$$\begin{cases}\dot{x}=-v_1\cdot\sin(\theta)\\ \dot{y}=v_1\cdot\cos(\theta)-v_2\\ \dot{\theta}=-u\\ \dot{v}_1=0\\ \dot{v}_2=0\end{cases} \tag{35}$$

为了控制飞机 1 不与飞机 2 发生碰撞, SafeNC 需要生成一个神经网络控制器$\mathcal{N}$, 以控制飞机 1 的飞行方向. 控制器$\mathcal{N}$接收系统当前的状态 $s=[x,y,\theta,v_1,v_2]$并输出控制信号 $u=\mathcal{N}(s)$. 由于系统状态包含 5 个状态变量, 所以神经网络控制器的输入为 5 维. 类似地, 其输出为一维, 表示控制信号 u 的值. 依照相关工作中对于此例子的设定, SafeNC 目标为生成一个结构为 5-50-50-50-50-50-50-1 的控制器, 其中, 每层的神经元数量用"-"分隔. 隐藏层中的激活函数使用 ReLU. SafeNC 同时学习一个神经网络作为障碍证书, 用以保证控制器的安全性. 其结构设置为 5-10-10-10-1, 激活函数使用 ReLU.

在随机初始化神经网络参数后, SafeNC 从系统的不安全区域、初始区域以及系统空间采集数据点构造训练数据集. 每个区域随机采集 10 万个数据点. 最大迭代次数设置为 50. 在首轮迭代中, 控制器在学习模块中经过 28.13 s 训练后, 在训练数据集上满足安全性. 此时, 学习模块终止并转入验证模块, 以检验候选控制器是否在整个系统上安全. 由于此系统为连续动态系统, 因此验证模块只需验证定义 6 中前 3 个障碍证书条件. 验证模块验证控制器是满足前两个障碍证书条件的, 然而在验证第 3 个条件时返回了反例. 验证模块花费时间为 42.09 s. SafeNC 在反例点周围的 0.01 邻域中随机采取 1 000 个数据点, 并将其加入训练数据集 D_3,以继续训练控制器. 在第 5 次迭代时, SafeNC 成功生成一个得到安全验证的神经网络控制器, 所花费时间为 227.96 s. 该案例表明了 SafeNC 在连续动态系统上生成安全神经网络控制器的有效性.

例 2(Darboux 混成系统[41]): 图 4 展示了 Darboux 混成系统. 系统包含两个模态, 每个模态对应于一个连续动态系统. 系统状态 s 为二维, 由变量$[x_1,x_2]$描述. 模态 M_1 的系统区域$\Psi_1=\{-4\leqslant x_1\leqslant 0,-4\leqslant x_2\leqslant 4\}$. 模态 M_2 的系统区域$\Psi_2=\{0\leqslant x_1\leqslant 4,-4\leqslant x_2\leqslant 4\}$. 每个模态都包含一个圆形的跳转区域 T 可以转换到另一个模态, 同时有相应的重置函数 R 作用于跳转时的系统状态变量. 其中:

- 由 M_1 跳转到 M_2 的跳转区域 $t_{1,2}=\{x_1^2+x_2^2\leqslant 0.5625\}$, 重置函数 $r_{1,2}$ 定义为

$$\begin{cases}x_1'=-x_1\\ x_2'=x_2\end{cases} \tag{36}$$

- 由 M_2 跳转到 M_1 的跳转区域 $t_{2,1}=\{x_1^2+x_2^2\leqslant 0.25\}$, 重置函数 $r_{2,1}$ 定义为

$$\begin{cases}x_1'=x_1-2\\ x_2'=x_2+1\end{cases} \tag{37}$$

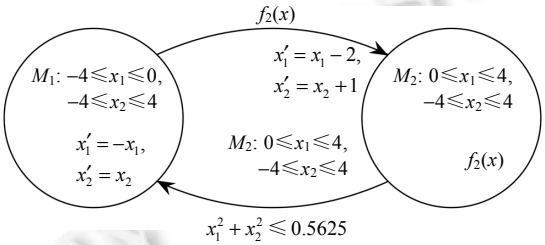

图 4 Darboux 系统

系统在两个模态上的演化方程 f 定义如下:

$$\begin{cases} f_1(x)=\begin{bmatrix}-x_1+u_1\\ -x_2\end{bmatrix} \\ f_2(x)=\begin{bmatrix}-x_1+2x_1^2x_2\\ u_2\end{bmatrix}\end{cases} \tag{38}$$

如公式(38)所示, 系统在每个模态都有一个控制器 u_i 控制系统演化. 系统轨迹从模态 M_1 出发, 其初始区域定义为

$$I_1=\{(x_1+2)^2+(x_2-2)^2\leqslant 0.25\} \tag{39}$$

不安全区域为在模态 M_2 上的给定区域, 定义为

$$U_2=\{(x_1-2)^2+(x_2-2)^2\leqslant 0.25\} \tag{40}$$

SafeNC 需要为每个模态学习一个神经网络控制器, 保证从初始区域出发的系统轨迹不会进入不安全区域. 每个模态上的控制器的结构都相同. SafeNC 首先尝试生成使用 ReLU 激活函数的结构为 2-20-1 的小型神经网络控制器. 从系统的不安全区域、初始区域、系统空间以及跳转区域采样的训练数据集分别包含 10 万个数据点. 最大迭代次数设置为 50. 经过 4 次迭代, SafeNC 成功生成两个安全的神经网络控制器$\mathcal{N}_1$与$\mathcal{N}_2$, 运行时间为 53.62 s. 为了充分测试 SafeNC 的能力, 尝试生成结构为 2-200-200-200-200-200-1 的大型神经网络控制器. SafeNC 在经过 7 次迭代之后, 成功生成此结构控制器, 运行时间为 369.49 s. 两种不同结构控制器的成功生成, 表明了 SafeNC 在混成系统上生成安全神经网络控制器的有效性与扩展性.

4.3 实验评估

为了充分评估 SafeNC 的能力, 本文在 2 个混成系统和 6 个连续动态系统上对其进行测试, 并与两份同样基于障碍证书生成安全神经网络控制器的相关工作 nncontroller[14]和 SRLBC[15]做对比. nncontroller 使用 $\text{Bent-ReLU}: y=0.5x+\sqrt{0.25x^2+0.0001}$ 作为神经网络激活函数, 用以学习候选控制器以及障碍证书, 并使用可满足性模理论(SMT)求解器 iSAT3[42]对候选控制器进行安全验证. SRLBC 通过强化学习过程训练候选神经网络控制器, 并基于多项式抽象和双线性矩阵不等式求解对候选控制器生成多项式形式的障碍证书, 以进行安全验证. 实验评估结果见表 1.

表 1 实验评估以及与相关工作对比

ID	Structure of $\mathcal{N}$	SafeNC				nncontroller		SRLBC	
		$\mathcal{B}$	t_l (s)	t_v (s)	t (s)	$\mathcal{B}$	t (s)	deg$\mathcal{B}$	t (s)
H1[41]	2-200-200-200-200-200-1	2-50-50-1	114.84	254.65	**369.49**	–	–	–	–
H2[43]	2-80-40-20-1	2-30-1	69.53	145.22	**214.75**	–	–	–	–
C1[14]	2-20-1	2-20-1	20.46	27.15	47.61	2-10-1	54.62	2	**31.57**
C2[44]	2-50-50-1	2-20-1	34.95	31.63	**66.58**	2-20-1	237.59	2	69.42
C3[44]	2-100-50-20-1	2-50-1	57.91	125.58	183.49	2-50-1	383.16	2	**153.49**
C4[13]	3-100-100-100-100-1	3-20-20-1	92.75	179.21	**271.96**	3-20-1	1 637.52	4	397.28
C5[13]	3-300-300-300-300-300-300-1	3-50-30-10-1	166.26	406.92	**573.18**	3-100-1	4 917.38	4	1 349.16
C6[40]	5-50-50-50-50-50-50-1	5-10-10-10-1	83.15	144.81	**227.96**	–	–	4	1 946.87

在表 1 中, 以“H”开头的测试用例 ID 代表混成系统, 以“C”开头的代表连续动态系统. 对于每个测试用例, Structure of $\mathcal{N}$一列显示了目标神经网络控制器的结构. 在 SafeNC 栏目中, $\mathcal{B}$代表神经网络障碍证书的结构, t_l代表学习模块运行时所花费的时间, t_v代表验证模块花费的时间, t 代表成功生成被安全验证的控制器所花费的总时间. nncontroller 栏目展示了神经网络障碍证书的结构与运行总时间. SRLBC 栏目展示了多项式障碍证书的次数以及运行总时间. 表 1 中, 运行时间的单位为 s, 且数据为 10 次运行的平均值. 允许的最大运行时间为 36 000 s. “–”表示方法在用例上失败, 即不能在最大允许时间内生成安全的神经网络控制器.

观察表 1 可以发现, SafeNC 成功为所有测试用例生成安全的神经网络控制器, 包括 2 个混成系统和 6 个连续动态系统. SafeNC 可以生成含有 1 个隐藏层且总神经元数为 23 的小型神经网络控制器(C1), 同时也可以生成包含 6 个隐藏层并具有 1 804 个神经元的大型神经网络控制器(C5). 此外, SafeNC 可以同时生成多层的神经网络障碍证书, 用以辅助验证(C4–C6). 这些用例上的实验表明了 SafeNC 的有效性和扩展性.

对于 nncontroller 与 SRLBC, 它们只能为连续动态系统而不能为混成系统生成神经网络控制器, 成功的用例个数分别为 5 个以及 6 个. 对于 C6, nncontroller 成功地生成了候选控制器, 但是其使用 SMT 求解器对候选控制器验证时, 不能在 36 000 s 内返回验证结果. 观察 3 个工具的平均运行时间可以发现, SafeNC 在大部分用例上用时最短, 尤其是当生成大型神经网络控制器时(C4–C6). SafeNC、nncontroller、SRLBC 在所有例子上的平均运行时间分别为 244.38 s、1 446.05 s、657.97 s. 这些实验结果表明了 SafeNC 与现有方法相比的有效性和高效性.

- 实验分析

对于表 1 中 3 种方法的实验结果, 本文从方法框架、障碍证书模板、障碍证书生成过程以及安全验证方法这 4 个方面进行分析, 讨论 SafeNC 与 nncontroller 以及 SRLBC 相比的优势.

- 方法框架: SafeNC 与 SRLBC 的整体方法框架都为迭代反馈框架. 若在前一次迭代中对于候选控制器的验证没有成功, 那么这两种方法都会基于此次失败改进下一次迭代中候选控制器的训练过程. 如对于 SafeNC 即为验证模块会返回反例用于下一次迭代. 而 nncontroller 没有类似的迭代反馈过程.
- 障碍证书模板: SafeNC 与 nncontroller 采用神经网络作为障碍证书的生成模板, 而 SRLBC 使用多项式作为障碍证书的模板. 由于神经网络的高灵活性与对于连续函数的高拟合性, 神经网络障碍证书的表现往往要优于多项式障碍证书.
- 障碍证书生成过程: SafeNC 与 nncontroller 的障碍证书是同时与控制器进行训练学习的, 而 SRLBC 的障碍证书生成与控制器生成则是分开的. 当生成障碍证书时, 控制器固定不变, 反之亦然. 因此, SafeNC 与 nncontroller 的障碍证书学习过程更能对安全控制器的生成起到积极的作用.
- 安全验证方法: SafeNC 将控制器的安全验证问题转化为一组混合整数规划问题, 并使用数值优化器求解. nncontroller 则是使用 SMT 求解器直接求解原始安全验证问题. 而 SRLBC 则是求解固定模板的多项式障碍证书以进行安全验证. 与其他两种方法相比, SafeNC 的验证方法更灵活且高效.

5 相关工作

本文工作与对于混成系统的安全验证与安全控制的文献相关.

5.1 混成系统安全验证

混成系统的安全验证可以通过计算系统轨迹可达集或其过近似进行[40,45,46]. Claviere 等人[40]将验证模拟与抽象解释结合起来推断系统可达集. Hu 等人[46]提出了一种基于反馈互联的半正定算法, 对线性时变系统进行可达性分析. 为了缓解由于神经网络控制器导致的计算复杂度, 一些方法首先将原始系统转化为不含有神经网络控制器的系统, 然后再使用可达集计算方法对其进行安全验证[47−50]. Ivanov 等人[48]提出了一种利用泰勒模型将神经网络转化为等价的混成系统的方法. Sun 等人[49]对混成系统进行有限状态抽象, 并使用可达性分析技术以推断其安全的可达集. Fan 等人[51]使用伯恩斯坦多项式逼近具有不同激活函数的神经网络控制器, 以进行后续的可达集计算. 转换或者抽象神经网络控制器的方法会不可避免地带来误差, 从而损失验证的精确性. 此外, 由于对于神经网络控制器以及系统可达集计算是一步步进行的, 其计算过程中的误差会快速累积, 因此这些方法容易产生误报.

另一类方法将神经网络控制器从混成系统中分离出来, 并分别对控制器以及动力学模型两部分进行分析, 其安全验证问题转换为对神经网络控制器在给定输入范围内的输出范围验证问题及可达集计算问题[52]. 对于神经网络控制器的验证方法包括抽象解释[53−55]、区间算法[56]、线性逼近[57,58]和混合整数规划[59]. 然而, 由于此类方法简化了神经网络控制器和动力学模型之间的交互关系, 其验证的精度以及准确性会有较大损失. 此外, 对于动力学模型的验证仍然基于可达集计算技术, 其只能验证系统在有限时间范围内的安全性.

障碍证书可以保证混成系统在无限时间范围上的安全性[34,60,61]. Tuncali 等人[62]基于模拟运行为混成系统生成神经网络障碍证书. Zhao 等人[63]将 Bent-ReLU 函数作为神经网络障碍证书的激活函数, 并基于 SMT 求解器 iSAT3[42]对系统进行验证. Peruffo 等人[64]提出了一种基于反例制导的自动顺序循环方法来生成候选障碍证

书, 并使用 SMT 求解器 dReal[65]和 Z3[66]来验证它们. Sha 等人[5]用多项式逼近神经网络控制器, 从而合成障碍证书. 我们之前的工作[67]提出了一种基于神经网络障碍证书对连续动态系统进行安全验证的方法. 与之相比, 本文方法可以为混成系统生成安全的神经网络控制器与障碍证书. 此外, 本节所述工作的一个重要不足是: 它们只为系统提供安全验证, 而不参与安全的控制器以及混成系统的构建.

5.2 混成系统安全控制

基于机器学习为混成系统生成安全的神经网络控制器方法已经得到了广泛的研究[16−18]. 一类工作考虑系统发生不安全行为的可能性, 并将其编码到损失函数中进行优化, 以降低危险行为发生的概率[19−21]. Dutta 等人[68]提出了一种数据驱动的反馈方法, 将安全控制器学习和验证结合起来. Jin 等人[2]基于安全风险以构造学习优化目标, 并通过最小化损失以学习安全控制器. Tamar 等人[20]基于抽样技术提出了两种优化不安全风险损失的梯度更新方法. 然而, 由于此类方法是通过损失函数优化来保证混成系统的安全性, 它们只能减少而无法完全避免不安全行为的发生.

为了完全避免不安全行为, 另一类工作使用如控制障碍证书、控制李雅普诺夫函数以及安全的备用控制器技术来保证所生成的神经网络控制器的安全性[25−27]. Cheng 等人[69]结合控制障碍证书、无模型和有模型的控制器理论以及未知系统动力学生成技术, 提出了一种端到端的安全系统学习架构. Berkenkamp 等人[70]将控制李雅普诺夫函数理论扩展到混成系统动力学模型. Luo 等人[30]基于安全的备用策略以生成具有零训练时间的安全控制器. 上述方法基于额外的安全保证技术以生成安全的控制器. 然而, 控制障碍证书、控制李雅普诺夫函数以及安全的备用控制器通常需要人为预先提供, 这极大地限制了这些方法在混成系统中的应用. 与此类方法相比, 本文方法的安全保证是由同时与控制器一起训练的障碍证书提供的, 因而不需要任何先验知识.

文献[13−15]是与本文最相近的 3 个生成安全神经网络控制器的相关工作. 在文献[13]中, Deshmukh 等人基于提前给定的控制障碍证书构造了一种循环学习验证算法, 以生成候选的神经网络控制器, 并使用 SMT 求解器 dReal[64]进行验证. 与文献[13]相比, 本文的安全控制器生成过程不需要提前给定的先验知识; 并且, 本文将候选控制器安全验证问题转化为一组等价的优化问题, 然后使用数值优化器而不是 SMT 求解器进行求解, 在效率上有极大的提高. 文献[14,15]为本文的对比工作, 在其中, Zhao 等人和 Yang 等人分别提出了 nncontroller 和 SafeNC 工具. 如在第 4.3 节所讨论, 与文献[14,15]相比, 本工作在方法框架、障碍证书模板、障碍证书生成过程以及安全验证方法方面具有显著优势, 更高效且更具扩展性.

6 总 结

本文基于障碍证书提供的安全保证与验证反例提供的安全指导, 提出了一种为混成系统生成安全神经网络控制器的反馈驱动方法. 此方法由学习模块和验证模块组成. 学习模块首先确定控制器采用的神经网络模板结构, 基于系统构建训练数据集; 然后编码损失函数以训练控制器, 并保证其在数据集上的安全性. 验证模块将候选控制器的安全验证问题转化为一组混合整数规划问题, 并使用数值优化器进行求解, 以检验候选控制器在整个系统上是否安全. 我们实现了一个安全神经网络控制器生成工具 SafeNC, 并在 8 个测试用例上评估其性能. SafeNC 成功生成包含 6 个隐藏层且具有 1 804 个神经元的安全控制器, 其在所有用例上的平均运行时间为 244.38 s. 实验结果表明, 本文方法比现有方法更有效且更具扩展性.

References:

[1] Maasoumy M, Pinto A, Sangiovanni-Vincentelli A. Model-based hierarchical optimal control design for HVAC systems. In: Proc. of the Dynamic Systems and Control Conf. 2011. 271−278. [doi: 10.1115/DSCC2011-6078]

[2] Jin WX, Wang ZR, Yang ZR, *et al*. Neural certificates for safe control policies. arXiv:2006.08465, 2020.

[3] Taylor A, Singletary A, Yue YS, *et al*. Learning for safety-critical control with control barrier functions. In: Proc. of the Learning for Dynamics and Control. 2020. 708−717.

[4] Psaltis D, Sideris A, Yamamura AA. A multilayered neural network controller. IEEE Control Systems Magazine, 1988, 8(2): 17−21. [doi: 10.1109/37.1868]

[5] Sha M, Chen X, Ji YZ, *et al*. Synthesizing barrier certificates of neural network controlled continuous systems via approximations. In: Proc. of the 58th ACM/IEEE Design Automation Conf. (DAC). 2021. 631−636. [doi: 10.1109/DAC18074.2021.9586327]

[6] Bastani O. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In: Proc. of the American Control Conf. (ACC). 2021. 3488−3494. [doi: 10.23919/ACC50511.2021.9483182]

[7] Zhao QY, Chen X, Zhang YF, *et al*. Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems. In: Proc. of the 24th Int'l Conf. on Hybrid Systems: Computation and Control. 2021. 1−11. [doi: 10.1145/3447928.3456638]

[8] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 1989, 2(4): 303−314. [doi: 10.1007/BF02551274]

[9] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks, 1989, 2(5): 359−366. [doi: 10.1016/0893-6080(89)90020-8]

[10] Liu J, Zhan NJ, Zhao HJ. Computing semi-algebraic invariants for polynomial dynamical systems. In: Proc. of the 9th ACM Int'l Conf. on Embedded Software. 2011. 97−106. [doi: 10.1145/2038642.2038659]

[11] Sankaranarayanan S. Automatic invariant generation for hybrid systems using ideal fixed points. In: Proc. of the 13th ACM Int'l Conf. on Hybrid Systems: Computation and Control. 2010. 221−230. [doi: 10.1145/1755952.1755984]

[12] Xiang WM, Johnson TT. Reachability analysis and safety verification for neural network control systems. arXiv:1805.09944, 2018.

[13] Deshmukh JV, Kapinski JP, Yamaguchi T, *et al*. Learning deep neural network controllers for dynamical systems with safety guarantees. In: Proc. of the IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD). 2019. 1−7. [doi: 10.1109/ICCAD45719.2019.8942130]

[14] Zhao HJ, Zeng X, Chen TL, *et al*. Learning safe neural network controllers with barrier certificates. Formal Aspects of Computing, 2021, 33(3): 437−455. [doi: 10.1007/s00165-021-00544-5]

[15] Yang ZF, Zhang YD, Lin W, *et al*. An iterative scheme of safe reinforcement learning for nonlinear systems via barrier certificate generation. In: Proc. of the Int'l Conf. on Computer Aided Verification. 2021. 467−490. [doi: 10.1007/978-3-030-81685-8_22]

[16] Yang YL, Vamvoudakis KG, Modares H. Safe reinforcement learning for dynamical games. Int'l Journal of Robust and Nonlinear Control, 2020, 30(9): 3706−3726. [doi: 10.1002/rnc.4962]

[17] Thomas PS. Safe reinforcement learning [Ph.D. Thesis]. Amherst: University of Massachusetts Amherst, 2015. [doi: 10.7275/7529913.0]

[18] Garcıa J, Fernández F. A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research, 2015, 16(1): 1437−1480.

[19] Stooke A, Achiam J, Abbeel P. Responsive safety in reinforcement learning by pid lagrangian methods. In: Proc. of the Int'l Conf. on Machine Learning. 2020. 9133−9143.

[20] Tamar A, Chow Y, Ghavamzadeh M, *et al*. Sequential decision making with coherent risk. IEEE Trans. on Automatic Control, 2016, 62(7): 3323−3338. [doi: 10.1109/TAC.2016.2644871]

[21] Thananjeyan B, Balakrishna A, Nair S, *et al*. Recovery RL: Safe reinforcement learning with learned recovery zones. IEEE Robotics and Automation Letters, 2021, 6(3): 4915−4922. [doi: 10.1109/LRA.2021.3070252]

[22] Yang TY, Rosca J, Narasimhan K, *et al*. Accelerating safe reinforcement learning with constraint-mismatched baseline policies. In: Proc. of the Int'l Conf. on Machine Learning. 2021. 11795−11807.

[23] Srinivasan K, Eysenbach B, Ha S, *et al*. Learning to be safe: Deep RL with a safety critic. arXiv:2010.14603, 2020.

[24] Bharadhwaj H, Kumar A, Rhinehart N, *et al*. Conservative safety critics for exploration. arXiv:2010.14497, 2020.

[25] Alshiekh M, Bloem R, Ehlers R, *et al*. Safe reinforcement learning via shielding. In: Proc. of the AAAI Conf. on Artificial Intelligence. 2018. [doi: 10.1609/aaai.v32i1.11797]

[26] Ohnishi M, Wang L, Notomista G, *et al*. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. IEEE Trans. on Robotics, 2019, 35(5): 1186−1205. [doi: 10.1109/TRO.2019.2920206]

[27] Cohen MH, Belta C. Approximate optimal control for safety-critical systems with control barrier functions. In: Proc. of the 59th IEEE Conf. on Decision and Control (CDC). 2020. 2062−2067. [doi: 10.1109/CDC42340.2020.9303896]

[28] Xu XR, Tabuada P, Grizzle JW, *et al*. Robustness of control barrier functions for safety critical control. IFAC-PapersOnLine, 2015, 48(27): 54−61. [doi: 10.1016/j.ifacol.2015.11.152]

[29] Primbs JA, Nevistić V, Doyle JC. Nonlinear optimal control: A control Lyapunov function and receding horizon perspective. Asian Journal of Control, 1999, 1(1): 14−24. [doi: 10.1111/j.1934-6093.1999.tb00002.x]

[30] Luo YP, Ma T. Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations. In: Advances in Neural Information Processing Systems. 2021. 25621−25632.

[31] Ravanbakhsh H, Sankaranarayanan S. Learning control lyapunov functions from counterexamples and demonstrations. Autonomous Robots, 2019, 43(2): 275−307. [doi: 10.1007/s10514-018-9791-9]

[32] Khansari-Zadeh SM, Billard A. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. Robotics and Autonomous Systems, 2014, 62(6): 752−765. [doi: 10.1016/j.robot.2014.03.001]

[33] Wang L, Theodorou EA, Egerstedt M. Safe learning of quadrotor dynamics using barrier certificates. In: Proc. of the IEEE Int'l Conf. on Robotics and Automation (ICRA). 2018. 2460−2465. [doi: 10.1109/ICRA.2018.8460471]

[34] Prajna S, Jadbabaie A. Safety verification of hybrid systems using barrier certificates. In: Proc. of the Int'l Workshop on Hybrid Systems: Computation and Control. 2004. 477−492. [doi: 10.1007/978-3-540-24743-2_32]

[35] Tjeng V, Xiao K, Tedrake R. Evaluating robustness of neural networks with mixed integer programming. arXiv:1711.07356, 2017.

[36] Kearfott RB, Kreinovich V. Applications of interval computations: An introduction. In: Proc. of the Applications of Interval Computations. 1996. 1−22. [doi: 10.1007/978-1-4613-3440-8_1]

[37] Gurobi OLLC. Gurobi Optimizer Reference Manual. 2020. https://www.gurobi.com/documentation/current/refman/index.html

[38] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436−444. [doi: 10.1038/nature14539]

[39] Abadi M, Barham P, Chen JM, *et al*. TensorFlow: A system for large-scale machine learning. In: Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI 2016). 2016. 265−283.

[40] Clavière A, Asselin E, Garion C, *et al*. Safety verification of neural network controlled systems. In: Proc. of the 51st Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks Workshops (DSN-W). 2021. 47−54. [doi: 10.1109/DSN-W52860.2021.00019]

[41] Zeng X, Lin W, Yang ZF, *et al*. Darboux-type barrier certificates for safety verification of nonlinear hybrid systems. In: Proc. of the 13th Int'l Conf. on Embedded Software. 2016. 1−10. [doi: 10.1145/2968478.2968484]

[42] Scheiber K. iSAT3 Manual. 2017. https://projects.informatik.uni-freiburg.de/projects/isat3/

[43] Xue B, Fränzle M, Zhao HJ, *et al*. Probably approximate safety verification of hybrid dynamical systems. In: Proc. of the Int'l Conf. on Formal Engineering Methods. 2019. 236−252. [doi: 10.1007/978-3-030-32409-4_15]

[44] Zhu H, Xiong ZK, Magill S, *et al*. An inductive synthesis framework for verifiable reinforcement learning. In: Proc. of the 40th ACM SIGPLAN Conf. on Programming Language Design and Implementation. 2019. 686−701. [doi: 10.1145/3325983]

[45] Akametalu AK, Fisa JF, Gillula JH, *et al*. Reachability-based safe learning with Gaussian processes. In: Proc. of the 53rd IEEE Conf. on Decision and Control. 2014. 1424−1431. [doi: 10.1109/CDC.2014.7039601]

[46] Hu HM, Fazlyab M, Morari M, *et al*. Reach-SDP: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In: Proc. of the 59th IEEE Conf. on Decision and Control (CDC). 2020. 5929−5934. [doi: 10.1109/CDC42340.2020.9304296]

[47] Huang C, Fan J, Li W, *et al*. ReachNN: Reachability analysis of neural-network controlled systems. ACM Trans. on Embedded Computing Systems (TECS), 2019, 18(5s): 1−22. [doi: 10.1145/3358228]

[48] Ivanov R, Carpenter TJ, Weimer J, *et al*. Verifying the safety of autonomous systems with neural network controllers. ACM Trans. on Embedded Computing Systems (TECS), 2020, 20(1): 1−26. [doi: 10.1145/3419742]

[49] Sun XW, Khedr H, Shoukry Y. Formal verification of neural network controlled autonomous systems. In: Proc. of the 22nd ACM Int'l Conf. on Hybrid Systems: Computation and Control. 2019. 147−156. [doi: 10.1145/3302504.3311802]

[50] Xiang WM, Tran HD, Yang XD, *et al*. Reachable set estimation for neural network control systems: A simulation-guided approach. IEEE Trans. on Neural Networks and Learning Systems, 2020, 32(5): 1821−1830. [doi: 10.1109/TNNLS.2020.2991090]

[51] Fan JM, Huang C, Chen X, *et al*. ReachNN*: A tool for reachability analysis of neural-network controlled systems. In: Proc. of the Int'l Symp. on Automated Technology for Verification and Analysis. 2020. 537−542. [doi: 10.1007/978-3-030-59152-6_30]

[52] Julian KD, Kochenderfer MJ. A reachability method for verifying dynamical systems with deep neural network controllers. arXiv:1903.00520, 2019. [doi: 10.48550/arXiv.1903.00520]

[53] Singh G, Gehr T, Mirman M, *et al*. Fast and effective robustness certification. In: Proc. of the NeurIPS. 2018.

[54] Gehr T, Mirman M, Drachsler-Cohen D, *et al*. AI2: Safety and robustness certification of neural networks with abstract interpretation. In: Proc. of the IEEE Symp. on Security and Privacy (SP). 2018. 3−18. [doi: 10.1109/SP. 2018.00058]

[55] Singh G, Gehr T, Püschel M, *et al*. An abstract domain for certifying neural networks. In: Proc. of the ACM on Programming Languages. 2019. 1−30. [doi: 10.1145/3291645]

[56] Wang SQ, Pei KX, Whitehouse J, *et al*. Formal security analysis of neural networks using symbolic intervals. In: Proc. of the 27th USENIX Security Symp. (USENIX Security 2018). 2018. 1599−1614.

[57] Wong E, Kolter Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In: Proc. of the Int'l Conf. on Machine Learning. 2018. 5286−5295.

[58] Zhang ZD, Liu J, Liu GJ, *et al*. Robustness verification of swish neural networks embedded in autonomous driving systems. IEEE Trans. on Computational Social Systems, Early Access, 2022. [doi: 10.1109/TCSS.2022.3179659]

[59] Lin W, Yang ZF, Chen X, *et al*. Robustness verification of classification deep neural networks via linear programming. In: Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition. 2019. 11418−11427.

[60] Gan T, Xia BC. Barrier certificate generation for safety verification of continuous systems for a bounded time. Ruan Jian Xue Bao/ Journal of Software, 2016, 27(3): 645−654 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4986.htm [doi: 10.13328/j.cnki.jos.004986]

[61] Shen MJ, Zeng ZB, Lin W, *et al*. Safety verification of stochastic continuous system using stochastic barrier certificates. Journal of Computer Applications, 2018, 38(6): 1737−1744 (in Chinese with English abstract). [doi: 10.11772/j.issn.1001-9081. 2017112824]

[62] Tuncali CE, Kapinski J, Ito H, Deshmukh JV. Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In: Proc. of the 55th Annual Design Automation Conf. 2018. 1−6. [doi: 10.1145/3195970.3199852]

[63] Zhao HJ, Zeng X, Chen TL, Liu ZM. Synthesizing barrier certificates using neural networks. In: Proc. of the 23rd Int'l Conf. on Hybrid Systems: Computation and Control. 2020. 1−11. [doi: 10.1145/3365365.3382222]

[64] Peruffo A, Ahmed D, Abate A. Automated and formal synthesis of neural barrier certificates for dynamical models. In: Proc. of the Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. 2021. 370−388. [doi: 10.1007/978-3-030-72016-2_20]

[65] Gao SC, Kong S, Clarke EM. dReal: An SMT solver for nonlinear theories over the reals. In: Proc. of the Int'l Conf. on Automated Deduction. 2013. 208−214. [doi: 10.1007/978-3-642-38574-2_14]

[66] Moura L, Bjørner N. Z3: An efficient SMT solver. In: Proc. of the Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. 2008. 337−340. [doi: 10.1007/978-3-540-78800-3_24]

[67] Zhao QY, Chen X, Zhao ZY, Zhang YF, Tang EY, Li XD. Verifying neural network controlled systems using neural networks. In: Proc. of the 25th ACM Int'l Conf. on Hybrid Systems: Computation and Control. 2022. 1−11. [doi: 10.1145/3501710.3519511]

[68] Dutta S, Jha S, Sankaranarayanan S, Tiwari A. Learning and verification of feedback control systems using feedforward neural networks. IFAC-PapersOnLine, 2018, 51(16): 151−156. [doi: 10.1016/j.ifacol.2018.08.026]

[69] Cheng R, Orosz G, Murray RM, Burdick JW. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In: Proc. of the AAAI Conf. on Artificial Intelligence. 2019. 3387−3395. [doi: 10.1609/aaai.v33i01.33013387]

[70] Berkenkamp F, Turchetta M, Schoellig A, Krause A. Safe model-based reinforcement learning with stability guarantees. In: Proc. of the NeurIPS. 2017.

附中文参考文献:

[60] 甘庭, 夏壁灿. 运用障碍证书验证连续系统的有界时间安全性. 软件学报, 2016, 27(3): 645−654. http://www.jos.org.cn/1000-9825/4986.htm [doi: 10.13328/j.cnki.jos.004986]

[61] 沈敏捷, 曾振柄, 林望, 等. 基于随机障碍验证的随机连续系统安全性验证. 计算机应用, 2018, 38(6): 1737−1744. [doi: 10.11772/j.issn.1001-9081. 2017112824]

赵庆晔(1997－), 男, 博士生, CCF 学生会员, 主要研究领域为嵌入式系统安全, 混成系统安全控制与验证.

王豫(1991－), 男, 博士, CCF 专业会员, 主要研究领域为程序分析, 软件缺陷预测.

李宣东(1963－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为软件工程, 形式化方法.