软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007657] [CSTR: 32375.14.jos.007657]

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

基于线性松弛与对偶优化的循环神经网络验证方法*

赵 亮, 杨成龙, 闫宗祥, 王小兵

(西安电子科技大学 计算机科学与技术学院, 陕西 西安 710126)

通信作者: 王小兵, E-mail: xbwang@mail.xidian.edu.cn

摘 要: 随着当今软硬件和人工智能技术的不断发展, 以神经网络为代表的智能模型在各行各业被广泛应用, 但同时也暴露出不少安全问题, 例如与鲁棒性相关的对抗样本问题. 因此, 通过形式化验证的方式来检测并保障神经网络的鲁棒性和安全性至关重要. 然而, 现有的神经网络验证工作主要面向以 ReLU 作为激活函数的前馈神经网络, 对于结构复杂、激活函数非线性的循环神经网络 (RNN) 的验证工作比较有限. 基于此现状, 提出一种基于线性松弛与对偶优化的循环神经网络验证方法, 以计算鲁棒半径的方式为这类网络模型提供严格的鲁棒性保障. 首先, 将 RNN 验证问题通过线性松弛编码为输出关于输入的优化问题. 考虑到网络的层级结构, 将该优化问题进行拉格朗日分解, 每个子问题中仅包含相邻的两层神经元, 子问题之间采用同一性约束进行关联. 随后, 构建其对偶问题并采用梯度上升算法来优化求解, 得到输出神经元的近似区间上下界, 其中解的有效性由对偶优化的性质所保障. 最后, 对输入扰动采用二分搜索的方式计算网络的鲁棒半径. 通过在不同结构的朴素 RNN 与较为复杂的长短期记忆网络 (LSTM) 上与已有的验证方法进行实验对比, 结果表明所提方法在朴素 RNN 上求解出的鲁棒半径上较已有方法提升了 17.7%–30.2%, 在 LSTM 中亦有 5.2%–10.2% 的提升.

关键词: 循环神经网络; 形式化验证; 鲁棒半径; 对偶优化; 近似求解

中图法分类号: TP18

中文引用格式: 赵亮, 杨成龙, 闫宗祥, 王小兵. 基于线性松弛与对偶优化的循环神经网络验证方法. 软件学报. http://www.jos.org.cn/1000-9825/7657.htm

英文引用格式: Zhao L, Yang CL, Yan ZX, Wang XB. Verification Method for Recurrent Neural Network Based on Linear Relaxation and Dual Optimization. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7657.htm

Verification Method for Recurrent Neural Network Based on Linear Relaxation and Dual Optimization

ZHAO Liang, YANG Cheng-Long, YAN Zong-Xiang, WANG Xiao-Bing

(School of Computer Science and Technology, Xidian University, Xi'an 710126, China)

Abstract: With the rapid advancement of modern hardware, software and artificial intelligence technologies, intelligent models represented by neural networks have been widely applied across various industries. However, these models also reveal significant safety problems, such as the robustness-related problem of adversarial examples. Therefore, ensuring and detecting the robustness and safety of neural networks through formal verification methods is of paramount importance. However, most existing research on neural network verification primarily targets feedforward neural networks with ReLU activation functions. In contrast, verification efforts for recurrent neural networks (RNN), which feature more complex structures and nonlinear activation functions, are relatively limited. To address this issue, this study proposes a verification method for RNNs based on linear relaxation and dual optimization. This method provides rigorous robustness guarantees by computing the robustness radius of these networks. First, the RNN verification problem is formulated as an optimization problem relating the network's output to its input through linear relaxation. Taking into account the hierarchical structure of the network, the optimization problem is decomposed using Lagrangian decomposition into subproblems, each involving only two adjacent neuron

* 基金项目: 国家自然科学基金 (61972301, 62192734); 陕西省重点研发计划 (2023-YBGY-229); 西安市科技计划 (22GXFW0025)
收稿时间: 2025-02-07; 修改时间: 2025-11-14; 采用时间: 2026-02-26; jos 在线出版时间: 2026-06-10

layers, with equality constraints linking these subproblems. Next, the dual problem is formulated and solved using a gradient ascent algorithm to obtain approximate upper and lower bounds for the output neurons, with the validity of the solution ensured by the properties of dual optimization. Finally, binary search is applied to compute the network's robustness radius with respect to input perturbations. By conducting experimental comparisons with existing verification methods on vanilla RNNs of different structures and more complex long short-term memory (LSTM) networks, results demonstrate that the proposed method achieves a 17.7% to 30.2% improvement in the robustness radius for vanilla RNNs and a 5.2% to 10.2% improvement for LSTM networks.

Key words: recurrent neural network (RNN); formal verification; robustness radius; dual optimization; approximate solving

随着计算机软硬件算力的飞速发展, 以深度学习[1]等技术为核心的人工智能模型正在各个应用领域逐步展现关键作用, 包括自动驾驶、航天和医疗[2,3]等对安全性要求极高的应用. 例如, 无人机防撞系统 ACAS[4]主要使用深度神经网络框架, 通过传感器输入的环境数据对本机的速度、方向及其他状态进行最优的判断与调整. 虽然神经网络模型已经具备拟合复杂的输入输出关系的能力, 并且在计算效率方面具有显著优势, 但在实际系统的测试与应用中仍然暴露出不少安全问题, 其中以对抗样本问题最为显著[5]. 以图像分类任务为例, 原本可以被神经网络正确分类的原始样本在添加微小的对抗扰动之后得到的样本会被错误分类, 但在人眼看来原始样本与对抗样本之间并无明显差异. 由于这种微小扰动难以察觉, 因此其攻击具有较强的隐蔽性和较大的危害性[6,7]. 因此, 为确保神经网络模型在安全关键应用中的鲁棒性, 迫切需要采用形式化验证的方式为这类模型的行为提供理论上的安全保障. 这包括计算模型的鲁棒性边界, 确保分类决策在该边界内保持稳定.

在上述背景下, 学术界开展了对神经网络形式化验证的研究[8,9], 主要面向鲁棒性的验证, 相关工作可分为两大类: 精确求解方法[10]和近似求解方法[11]. 精确求解方法一般应用于采用 ReLU 作为激活函数的神经网络之中, 利用其分段线性的特点, 在求解时将网络中的参数转化为神经元之间满足的约束, 实践当中一般通过可满足性模理论 (satisfiability modulo theories, SMT)[12,13]或者混合整数线性规划 (mixed integer linear programming, MILP)[14,15]等技术对这类问题进行处理. 这类方法的优势在于能够精确地表示网络中的分段线性项, 使得计算出的结果没有任何误差, 但其缺点也很明显, 即无法应用在规模较大的网络之中, 因为计算中描述激活函数的约束数量与神经元数量之间是指数级的关系. 而近似求解方法主要采用线性松弛[16−18]、半正定规划[19,20]等技术, 对于特定的扰动类型与扰动范围, 通过将激活函数松弛为包含其凸包的线性约束区域, 可以通过迭代回溯的方式逐层计算神经网络的近似区间, 并通过分析输出层的近似区间状态来判断神经网络是否具有鲁棒性. 这类方法由于在激活函数的抽象过程中引入了近似误差, 仅能求得近似的鲁棒性边界, 但优点在于效率高、能够扩展至较为复杂的网络模型.

目前, 神经网络安全性验证的工作大多面向的是前馈神经网络或者卷积神经网络[21], 以采用 ReLU 激活函数的分类网络为主. 相对而言, 关于循环神经网络 (recurrent neural network, RNN)[22,23]的验证工作却非常有限. 而循环神经网络模型已被广泛应用于序列处理包括自然语言处理等多种现实场景, 其鲁棒性的检验和保障亦具有重要意义. 实际上, RNN 的鲁棒性验证对比前馈神经网络更为复杂. 这是由于此类网络通常用来处理序列输入, 网络层数较深, 且使用 tanh 等 S 型激活函数而并非分段线性函数. 此外, 若要验证长短期记忆网络 (long short-term memory, LSTM) 等存在神经元变量复杂耦合的模型则难度更大. 鉴于上述挑战, 以 POPQORN[24]为代表的现有工作采用一种鲁棒半径求解的近似求解技术来计算 RNN 模型的鲁棒性边界. 鲁棒半径确保了一个安全的局部可扰动范围, 求解出的鲁棒半径越大, 则能够提供越高的安全保障. 然而现有方法受限于基于反向传播的一般近似框架中激活函数的近似方式, 采用一个线性上界与一个线性下界对非线性激活函数进行近似, 这种方式带来的近似误差较大, 而且在网络层数较深时, 误差的逐层放大会导致结果较为粗糙.

针对上述问题, 本文提出了一种基于线性松弛与对偶优化的 RNN 鲁棒性验证方法, 采用三角松弛减小激活函数的近似误差, 并利用拉格朗日分解对验证问题进行重构与优化, 以提升鲁棒半径求解的精确性. 首先, 对于朴素 RNN 中的 S 型激活函数设计了一种三角松弛与线性上下界相结合的近似方式, 提高了激活函数的近似精度. 由于三角松弛约束会导致反向符号区间传播的约束数量呈指数级增长, 因此方法未采用经典的近似求解框架; 对于 LSTM 网络中的二元非线性乘积项则提出一种基于采样的平面近似方式. 其次, 注意到网络验证问题本质上是一

个输出关于输入的优化问题, 本方法利用拉格朗日分解的思想通过中间层神经元的变量复制将原求解问题进行逐层分解, 形成一系列可以独立求解的子问题, 再将相邻层子问题的解通过同一性约束进行联系, 得到重新构建的优化问题. 在此基础上, 构建并求解其对偶优化问题来得到输出变量的区间范围. 其优势在于充分利用了网络的层次结构, 子问题的求解计算过程类似于神经网络训练过程中的前向传播与反向传播, 同层的子问题之间可以并行独立求解, 因此本方法同样适用于较深层的网络. 同时基于弱对偶性的保障, 对偶问题的任一个解总是原问题解的下界, 因此可以任选求解迭代次数, 从而在精度和求解效率之间进行平衡. 最后, 基于对网络输出区间的判断, 对输入扰动采用二分法来搜索计算模型的鲁棒半径. 本方法被实现为一套验证工具 VRNN-LD (verification of RNN based on linear relaxation and dual optimization), 并与现有的高效求解方法 POPQORN 进行了实验对比. 实验结果表明, VRNN-LD 在朴素 RNN 与 LSTM 模型的鲁棒半径求解中均呈现出明显的精度提升.

综上所述, 本文的主要贡献总结为以下 3 点.

● 提出一种新的 RNN 鲁棒半径计算方法, 将对偶优化技术应用于 RNN 验证, 通过拉格朗日分解将原问题进行重构, 并求解其对偶问题来获得输出层边界.

● 面向朴素 RNN 的特点, 对于 tanh 等 S 型激活函数采用了三角松弛与线性上下界结合的近似方式, 对于输入上界小于 0 或者下界大于 0 的激活前神经元采用三角松弛, 可有效提升近似精度; 对于 LSTM 网络中的二元非线性项提出一种基于采样的平面上下界近似方式, 保障曲面近似的精确性.

● 将方法实现为验证工具 VRNN-LD, 并在多种朴素 RNN 与 LSTM 模型以及扰动距离度量下进行实验, 其求得的鲁棒半径相较于现有方法分别提升了 17.7%–30.2% (朴素 RNN) 与 5.2%–10.2% (LSTM).

值得指出的是, 本文方法面向 RNN 验证问题, 该问题与以 ReLU 激活函数为主的前馈/卷积神经网络的验证问题有明显不同. 首先, RNN 验证需要处理 tanh 等非分段线性激活函数, 对其近似易产生较大误差, 近似精度至关重要. 针对此特性, 本文采用一种三角松弛与线性上下界相结合的近似方式, 精度较现有方法有所提升. 其次, RNN 验证需处理 LSTM 单元等门耦合复杂结构及其产生的二元非线性项. 针对此特性, 本文提出一种基于采样的平面近似方式, 以有效计算曲面的近似上下界.

本文第 1 节列出当前主流的神经网络鲁棒性验证方法. 第 2 节介绍本文涉及的基础知识与记号. 第 3 节给出朴素 RNN 的验证方法. 第 4 节展示 LSTM 模型的验证方法. 第 5 节列出相关实验结果并进行分析. 最后总结全文.

1 神经网络鲁棒性验证相关工作

现有的神经网络鲁棒性验证工作大致可以分为精确求解和近似求解两类. 精确求解的方法通常基于 SMT/MILP 编码来实现. Katz 等人[13]提出 SMT 求解器 Reluplex, 利用扩展单纯形 (simplex) 法支持 ReLU 约束, 并对基于 ReLU 的前馈神经网络的鲁棒性进行了验证, 但由于计算成本高, 导致可扩展性不足. Ehlers[25]对神经网络模型中的约束进行整体近似, 引入了整数算法, 将基于 SMT 的近似精度大大提高, 进而减少了求解器调用次数, 节省了时间. Narodytska 等人[26]则对二值神经网络 (binarized neural network, BNN) 的鲁棒性验证方法进行了探索, 利用布尔公式编码与布尔可满足性 (Boolean satisfiability, SAT) 求解方法为基础进行求解. 这种方法对深度神经网络进行了无近似的精确布尔表示, 通过研究 SAT 域中的属性来分析 BNN 的相关属性, 且保证精确的属性映射关系. 然而受限于 SAT 域, 该方法难以对其他复杂类型的神经网络进行建模和求解. 除了 SMT/SAT 相关的方法外, Cheng 等人[27]采用混合整数线性规划 (mixed integer linear programming, MILP) 求解器对神经网络的输出层边界进行求解, 对分段线性项利用分支定界法 (branch and bound, BaB) 进行处理, 采用 big-M 策略进行非线性表达式的线性化. 总体而言, 这类方法具有较高的复杂度, 主要适用于依赖分段线性激活函数的小型前馈神经网络.

近似求解的方法对网络中的非线性单元进行松弛和抽象, 能够简化计算复杂度, 但由于松弛的过程引入了误差, 通常只能获得一个近似解, 不保证对所有验证问题得到确定结果. 这类方法一般基于符号区间传播、抽象解释和凸松弛等技术. Wang 等人[28]设计提出了基于符号区间传播的 ReluVal 方法, 在区间计算过程中考虑逐层神经元的线性/非线性依赖关系, 求解得到的区间精度相较于传统区间上下界数值计算的方法有明显提升. 同时 Zhang 等

人[29]将符号区间传播与 ReLU 激活函数的线性近似相结合提出了 CROWN, 该框架具有较好的效率和可扩展性. 在后续的工作中 Wang 等人[30]、Zhang 等人[31]在此框架中继续提出 α,β-CROWN、GCP-CROWN 等方法, 通过结合分支定界法 (BaB) 和割平面法, 进一步提高了验证精度. 另一方面, 苏黎世理工学院 (ETH) 的研究人员 Singh 等人[16,32]将抽象解释的思想运用到了神经网络区间求解当中, 对每一层的神经元提供一个数值上下界和一个线性关系上下界, 将网络中的权重矩阵、激活函数、卷积算子和最大池化算子等用对应的条件仿射变换函数进行建模. 在神经元抽象时可以分别使用区间抽象域、zonotope 以及幂集抽象域对这些仿射函数进行计算分析, 最终得到输出层变量上下界与输入之间的线性约束关系, 典型方法包括 AI2[16]、DeepZ[32]等. 在后续工作中为了进一步提高精度和扩展性, Singh 等人[33]继而提出 kPoly, 在近似神经网络中的非线性项时同时考虑多个 ReLU 节点的整体近似输出而不是逐点单独计算, 利用变量相关性进一步提高抽象精度. 在基于凸松弛的近似研究中, Wong 等人[34]采用凸多面体松弛的方式处理网络非线性单元, 随后构建对应的线性规划问题并求解, 他们证明该线性规划的对偶问题可以表示为类似反向传播网络的层次结构, 从而利用并行性对问题进行高效求解, 其方法还可以被用来指导神经网络的训练, 得到鲁棒性更强的网络. 值得一提的是, Gowal 等人[35]基于对偶理论的思想, 用一个无约束的非凸优化问题表示网络边界的求解过程, 并采用拉格朗日松弛法进行原始变量的计算, 对偶变量的优化过程采用梯度下降法. 基于弱对偶性的特点, 任何对偶变量的选择都可以获得一个鲁棒边界的下界. Bunel 等人[36]在此基础上结合拉格朗日分解进一步提高了对前馈神经网络的边界计算精度并证明该方法得到的结果至少与 Gowal 等人[35]的方法同样紧密, 同时将方法扩展至卷积神经网络. 总体而言, 近似求解方法能够处理相对较大的网络模型, 若能提升精度则可有效求解更多的验证问题. 上述方法主要面向以 ReLU 激活函数为主的前馈神经网络或者卷积神经网络.

由于 RNN 结构较复杂且采用 tanh 等非分段线性激活函数, 面向 RNN 的验证工作相对前馈神经网络较少, 并且均采用近似求解. 具有代表性的是 Ko 等人[24]提出的 POPQORN 方法, 采用类似 CROWN 的反向传播框架, 通过对 tanh 激活函数采用平行近似的方法计算 RNN 的鲁棒半径, 该方法具有较好的扩展性和高效性, 可用于求解朴素 RNN 与 LSTM 等网络结构的鲁棒半径. 另一项相关研究工作是 Du 等人[37]提出的 CERT-RNN 方法. 该方法采用抽象解释中的 DeepZ 验证框架, 其输入抽象域为 zonotope, 并在此基础上改进了 tanh 激活函数的抽象变换过程, 支持 l_∞ 范数度量下 RNN 鲁棒半径的求解. 与上述工作不同, 本文所提出的方法对激活函数在输入处于不同区间下采用更为精确的线性近似方式, 求解方法上利用对偶优化问题求解等技术实现验证精度与效率的平衡, 同时支持在多种范数度量下有效地求解鲁棒半径.

2 基础知识

本文所提方法主要用于计算 RNN 模型的鲁棒半径, 本节将介绍其相关概念和基本知识以及符号定义.

2.1 循环神经网络定义

循环神经网络 (RNN) 是一种具有递归特点的网络结构, 网络中每一层神经元的输出同时依赖于上一层的输出以及当前层的输入, 这使得网络可以记住上一个输入序列的特征, 同时学习到序列样本之间的规律. 因此其常被用在自然语言处理、异常检测等领域, 本文主要研究对象为朴素 RNN 与 LSTM 两种常见的网络结构. 一个朴素 RNN 的主要计算过程可以用公式 (1) 进行表示:

$$\begin{cases} a^{(t)} = \tanh(W^{aa}a^{(t-1)} + W^{ax}x^{(t)} + b^{a}) \\ R(X) = W^{Fa}a^{(m)} + b^{F} \end{cases} \tag{1}$$

公式 (1) 为 RNN 隐藏层之间的状态传递以及输出层的计算公式, 其中 $a^{(t)}$ 代表第 t 层的隐藏层状态, W^{aa} 代表网络隐藏层前向传播的权重矩阵, $x^{(t)}$ 表示第 t 层的输入, W^{ax} 则表示网络隐藏层与输入之间的权重矩阵, b^{a} 为隐藏层偏置. 公式 (1) 同样说明了 RNN 每一时刻的神经元状态同时依赖于当前时刻的输入以及上一时刻的状态, 体现了 RNN 对历史信息的记忆能力. 由于神经网络鲁棒性验证问题一般针对分类网络, 这类网络只有最后一层拥有输出, 表示最终的分类标签, 因此本文在分析和实验中同样选取用于分类的 RNN 网络. 输出层计算公式中 m 代表 RNN 的层数, 用 R 代表整个网络模型与输入之间的运算函数, W^{Fa} 和 b^{F} 分别代表输出层参数, X 表示网络的一个

输入序列. 一个 2 层的朴素 RNN 结构由图 1 所示.

除了朴素 RNN 结构外, 长短期记忆网络 (LSTM) 也是一种常见的 RNN 结构. 朴素 RNN 在网络层数较深的场景中, 无法对更久之前的信息进行有效的记忆, 性能瓶颈较为明显. 而 LSTM 模型是一种特殊的以记忆单元为单位的 RNN 网络, 其中每一层的处理都包含更多的线性与非线性计算, 这种结构对历史信息进行了选择性遗忘或记忆, 能够在网络层数较深时仍表现出较好的性能. 图 2 给出了一个 LSTM 记忆单元的细节组成. 其主要计算过程如公式 (2) 所示:

$$\begin{cases} f^{(t)} = \sigma(W^{fx}x^{(t)} + W^{fa}a^{(t-1)} + b^{f}),\ g^{(t)} = \tanh(W^{gx}x^{(t)} + W^{ga}a^{(t-1)} + b^{g}) \\ i^{(t)} = \sigma(W^{ix}x^{(t)} + W^{ia}a^{(t-1)} + b^{i}),\ o^{(t)} = \sigma(W^{ox}x^{(t)} + W^{oa}a^{(t-1)} + b^{o}) \\ c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)},\ a^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \end{cases} \tag{2}$$

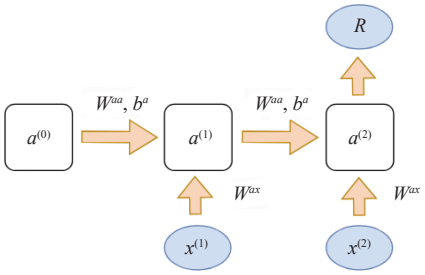

图 1 一个 2 层的朴素 RNN 示例

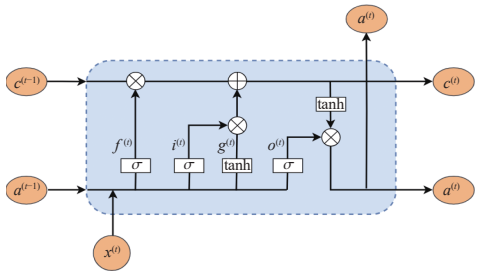

图 2 LSTM 记忆单元示意图

从图 2 中可以看出, LSTM 的中间层变量除了 $a^{(t)}$ 外, 还有一个记忆信息存储变量 $c^{(t)}$, 上一层的信息与当前层的输入由 4 个不同的门控单元遗忘门 $f^{(t)}$、输入门 $i^{(t)}$、输出门 $o^{(t)}$、候选门 $g^{(t)}$ 共同进行筛选与保留. 公式 (2) 中符号 W、x、b 的含义同朴素 RNN, 分别表示每个门控单元对应的权重矩阵、当前时刻输入以及偏置. 此处 σ 符号专指 Sigmoid 激活函数, $\odot$ 符号代表 Hadamard 积. 正是由于内部结构的复杂性, 所以其验证难度相较于朴素 RNN 也大大增加.

2.2 循环神经网络鲁棒性验证问题

首先给出验证过程中的相关符号与定义. 令 $X_0 = [x_0^{(1)}, \ldots, x_0^{(m)}]$ 为 RNN 的一个未经扰动的初始输入序列, 而 $X = [x^{(1)}, \ldots, x^{(m)}]$ 则表示初始输入经过一定扰动后得到的具体扰动序列. $label(X_0)$ 表示输入序列的标签值, 初始输入均能够被网络正确分类, 即 $label(X_0) = R(X)$. $R(X)$ 代表扰动序列通过循环神经网络计算处理后得到的输出结果.

鲁棒性验证问题定义: 固定距离度量与扰动范围 p 和 ε, 对于所有满足条件 $x^{(t)} \in B_p(x_0^{(t)}, \varepsilon)$ 的扰动输入 X, 是否均满足 $label(X_0) = R(X)$. $x^{(t)} \in B_p(x_0^{(t)}, \varepsilon)$ 表示 $x^{(t)}$ 满足 $\|x^{(t)} - x_0^{(t)}\|_p \leqslant \varepsilon$.

上述定义表明神经网络的鲁棒性验证问题就是对某个初始输入样本进行某种距离下的扰动, 在该扰动空间内的所有样本经过网络处理后是否都能得到和初始样本相同的分类结果. 如果能够全部正确分类说明网络在该情况下具备鲁棒性, 否则表示存在对抗样本.

鲁棒半径定义: 给定一个距离度量 p, 若扰动距离 ε 满足 $\forall x^{(t)} \in B_p(x_0^{(t)}, \varepsilon)$, $label(X_0) = R(X)$, 则称 ε 是网络的一个鲁棒半径.

从定义中可以得到, 鲁棒半径是一个具体的距离值, 表示在该距离内所有扰动样本均被正确分类, 鲁棒半径越大, 说明网络的抗干扰能力越强. 虽然在理论上鲁棒半径值存在一个上界. 然而, 文献 [13] 给出证明即使是对于采用 ReLU 的简单神经网络, 要确定其鲁棒半径上界也是一个 NP 完全问题. 对于采用其他非分段线性的激活函数的网络则更是无法精确求解. 因此近似求解的方法只能求解出小于其上界的鲁棒半径, 求解出的鲁棒半径大小可以作为评判近似求解方法精度的依据.

2.3 一般求解框架中的激活函数近似

RNN 中主要使用 tanh 与 Sigmoid 激活函数, 以往的鲁棒性验证工作中普遍采用符号区间传播的方式, 对于 tanh 的线性松弛采用一条上边界线与一条下边界线 (Sigmoid 与之类似), 如公式 (3) 所示:

$$\begin{cases} h_{U,r}^{(t)}(v) = \alpha_{U,r}^{(t)}(v+\beta_{U,r}^{(t)}) \\ h_{L,r}^{(t)}(v) = \alpha_{L,r}^{(t)}(v+\beta_{L,r}^{(t)}) \end{cases} \tag{3}$$

其中, $l_r^{(t)} \leqslant v \leqslant u_r^{(t)}$, $l_r^{(t)}$ 表示网络中第 t 层的第 r 个神经元经过激活前的近似区间下界而 $u_r^{(t)}$ 表示对应上界. 公式 (3) 给出两个简单的一元线性函数, 分别表示激活函数在给定输入范围内的近似上界线与下界线, 即对于 $l_r^{(t)} \leqslant v \leqslant u_r^{(t)}$, 有 $h_{L,r}^{(t)}(v) \leqslant \tanh(v) \leqslant h_{U,r}^{(t)}(v)$. 此外, 两个线性函数的斜率与截距根据 $l_r^{(t)}$ 与 $u_r^{(t)}$ 的大小得到, 即 $\alpha_{U,r}^{(t)}$、$\alpha_{L,r}^{(t)}$、$\beta_{U,r}^{(t)}$、$\beta_{L,r}^{(t)}$ 的值依赖于 $l_r^{(t)}$ 与 $u_r^{(t)}$. 根据输入范围的不同, 近似时用 3 种不同的方式来进行 tanh 上下界线的参数确定, 如图 3 所示.

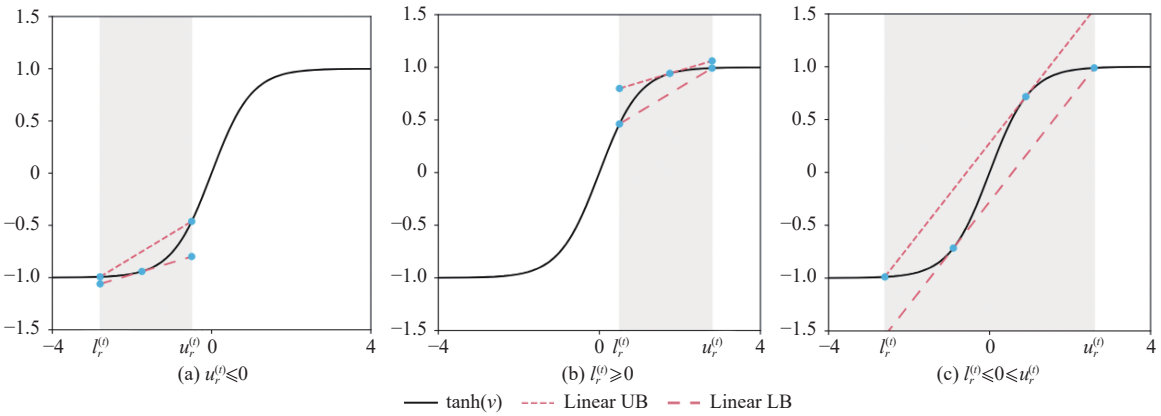

图 3 一般近似框架 tanh 激活函数近似方法

图 3(a) 表示 $u_r^{(t)} \leqslant 0$ 的情况, 此时上界线为左右两个端点的连线, 下界线为 $l_r^{(t)}$ 与 $u_r^{(t)}$ 范围内曲线上任意一点的切线; 图 3(b) 表示 $l_r^{(t)} \geqslant 0$ 的情况, 此时下界线为左右两个端点的连线, 上界线为 $l_r^{(t)}$ 与 $u_r^{(t)}$ 范围内曲线上任意一点的切线; 图 3(c) 表示 $l_r^{(t)} \leqslant 0 \leqslant u_r^{(t)}$ 的情况, 此时上界线过左端点与 $[l_r^{(t)}, u_r^{(t)}]$ 范围内曲线上某一点, 并且是该点的切线, 下界线过右端点与 $[l_r^{(t)}, u_r^{(t)}]$ 范围内曲线上某一点, 并且是该点的切线.

可以看到, 这种一条上界线与一条下界线的松弛方式使得非线性激活函数线性化, 求解中利用权重矩阵反向计算确定输出与输入之间的线性关系. Wu 等人[38]提出一种更细粒度的近似边界线求解方法, 分情况计算的整体思路不变, 主要在图 3(a) 和图 3(b) 的情况下限制了上下界线的斜率相同, 也称平行近似, 如图 4 所示. 该方法与图 3 所示近似方法相比, 可以有效避免某些情况下近似不够紧密的问题. 总体而言这一类近似方法受限于近似边界数量, 引入了较大的误差面积, 并且经过逐层传播会导致误差被进一步放大. 本文方法在求解朴素 RNN 鲁棒半径中针对这一问题进行了改进.

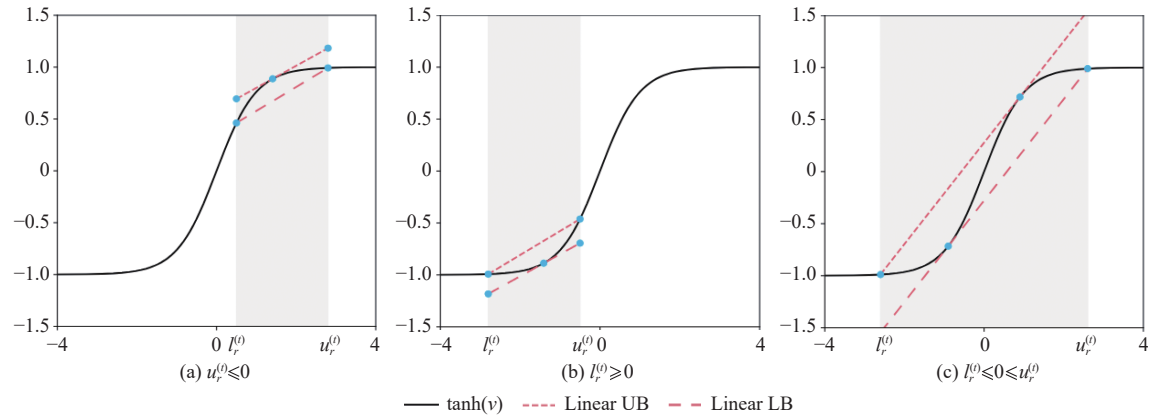

图 4 细粒度 tanh 激活函数近似方法

3 朴素 RNN 鲁棒半径求解

本文提出了一种基于线性松弛与对偶优化的 RNN 验证方法, 支持对朴素 RNN 与 LSTM 模型进行鲁棒半径求解. 本节介绍朴素 RNN 的鲁棒半径求解方法, 其总体流程如图 5 所示. 首先, 将 RNN 鲁棒性验证问题初始化为约束求解问题, 对其中的 tanh 激活函数采用三角近似与平行近似相结合的方法进行线性松弛. 随后, 将该优化问题进行拉格朗日分解, 得到一系列可以独立求解的子问题, 每个子问题只依赖于网络当中相邻的两层神经元, 并且子问题之间通过同一性约束进行关联, 保证整体解的一致性. 在此基础上, 构建并求解其对偶优化问题来得到输出变量的区间范围. 最后, 判断网络输出区间是否满足安全性质, 如果满足则利用二分法扩大扰动半径并再次计算, 反之缩小扰动半径, 直到精度达到预设的要求, 最终求得具体的鲁棒半径. 本节接下来就方法中的核心步骤作详细介绍, 最后总结出求解算法.

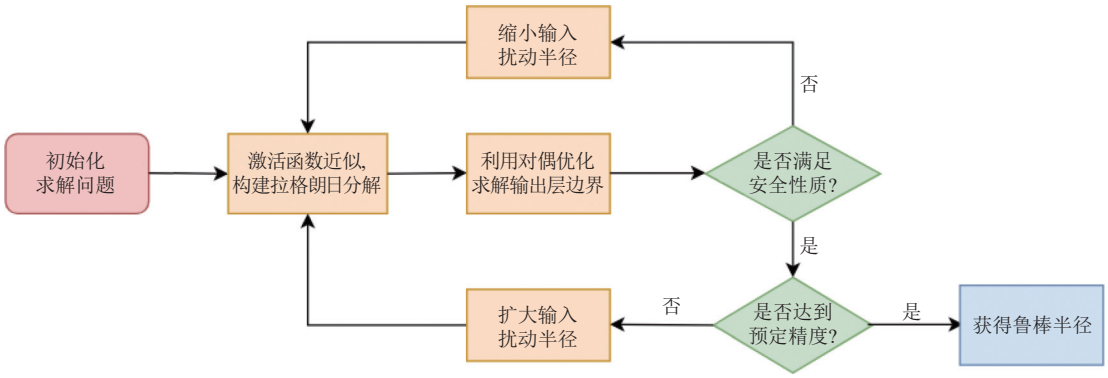

图 5 RNN 鲁棒半径求解总体流程

3.1 激活函数线性近似

RNN 主要采用 tanh 激活函数, 首先对其进行线性近似. 回顾第 2.3 节介绍了已有求解框架中的激活函数近似方法, 均采用一条上界线与一条下界线, 其近似误差较大. 与之不同的是, 本方法并不以反向传播技术为核心, 所以在对激活函数进行松弛时不必受限于近似上下界数量, 因此可以通过引入三角松弛来提高近似精度, 具体近似方法如图 6 所示. 其中, 图 6(a) 表示 $u_r^{(t)} \leqslant 0$, 此时上界线为左右两个端点的连线, 由于 tanh 函数在小于 0 的区间内是凸函数, 因此取 tanh 在左右两个端点 $l_r^{(t)}$ 与 $u_r^{(t)}$ 的切线作为线性下界, 两条下界线相交于一点 (i_1, j_1), 并且与上界线构成一个三角松弛区域; 图 6(b) 表示 $l_r^{(t)} \geqslant 0$, 此时下界线为左右两个端点的连线, 由于 tanh 函数在大于 0 的区间内是凹函数, 因此取 tanh 在左右两个端点 $l_r^{(t)}$ 与 $u_r^{(t)}$ 的切线作为线性上界, 两条上界线相交于一点 (i_2, j_2), 并且与下界线构成一个三角松弛区域; (c) 表示 $l_r^{(t)} \leqslant 0 \leqslant u_r^{(t)}$, 由于 tanh 的凹凸性在 0 附近发生变化, 此时采用与一般近似框架中相同的近似方式, 即上界线过左端点与 $[l_r^{(t)}, u_r^{(t)}]$ 范围内曲线上某一点, 并且是该点的切线. 下界线过右端点与 $[l_r^{(t)}, u_r^{(t)}]$ 范围内曲线上某一点, 并且是该点的切线. 上述松弛区域的约束 cvx_hull 可用公式 (4) 统一描述.

$$\text{cvx\_hull}(a^{(t)}, \hat{a}^{(t)}, l^{(t)}, u^{(t)}) \equiv \begin{cases} h_{L,r}^{(t)}(\hat{a}_r^{(t)}) \leqslant a_r^{(t)} \leqslant h_{U,r}^{(t)}(\hat{a}_r^{(t)}) \\ l_r^{(t)} \leqslant \hat{a}_r^{(t)} \leqslant u_r^{(t)} \end{cases} \tag{4}$$

公式 (4) 采用统一的方式表述了本方法的线性近似方式, 包括上述 3 种情形. 其中 $\hat{a}^{(t)}$ 和 $a^{(t)}$ 分别表示第 t 层神经元的输出在激活前和激活后的取值, $l^{(t)}$ 和 $u^{(t)}$ 分别为第 t 层神经元激活前值的近似下界和上界向量. 此外, $h_{L,r}^{(t)}(v)$ 在图 6(a) 情况下表示两条下边界与其交点构成的折线, 而在其他两种情况下表示单一线性下边界; $h_{U,r}^{(t)}(v)$ 与之类似, 在图 6(b) 情况下表示两条上边界与其交点构成的折线, 而在另外两种情况下表示单一线性上边界. 注意到松弛区域仅与中间变量激活前上下界有关, 从图 6 可以看出这种近似方式相较于已有方法有效地提升了激活函数的

近似精度, 且用于描述松弛的变量和约束的数量仅随网络中的单元数量线性增长. 在此基础上, 可对待验证问题进行拉格朗日分解并构建相应的对偶问题.

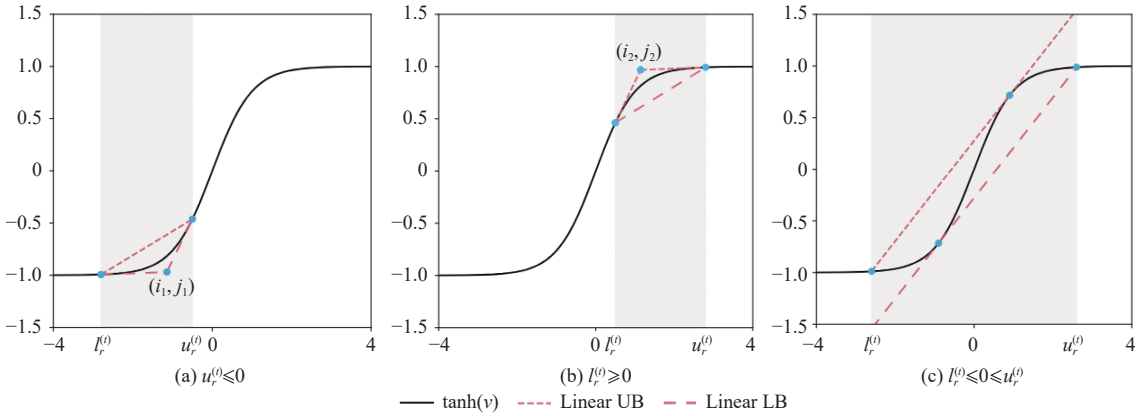

图 6 本方法中 tanh 激活函数近似方法

3.2 朴素 RNN 拉格朗日分解与对偶问题的构建

一般而言, 在求解朴素 RNN 鲁棒性验证问题时, 网络输入处于以原始样本为中心的球形区域内, 输出为每个标签置信度的近似区间. 因此, 可以将鲁棒性验证问题描述为网络输出最大最小值的约束求解问题. 本文对最小值的求解进行分析, 最大值同理. 具体而言, 对于一个 m 层的朴素 RNN 结构, 其鲁棒性验证问题可描述为公式 (5) 所示的优化问题. 回顾 $R(X)$ 表示网络输出, W^{Fa} 和 b^F 分别为输出层权重和偏置, W^{aa} 和 b^a 分别为隐藏层权重和偏置, W^{ax} 则表示网络隐藏层与输入之间的权重, $x^{(t)}$ 表示第 t 层的扰动输入, 取自初始输入 $x_0^{(t)}$ 在距离度量 p 下半径为 ε 的邻域.

$$\min_{a,\hat{a}} R(X), \text{ where } R(X) = W^{Fa}a^{(m)} + b^F, \text{ s.t. } \begin{cases} a^{(t)} = \tanh(\hat{a}^{(t)}),\ t \in [1,m] \\ \hat{a}^{(t)} = W^{aa}a^{(t-1)} + W^{ax}x^{(t)} + b^a,\ t \in [1,m] \\ x^{(t)} \in B_p(x_0^{(t)}, \varepsilon),\ t \in [1,m] \end{cases} \tag{5}$$

上述优化问题由于约束涉及多个层次的变量以及非线性激活函数的作用导致其难以求解. 为此, 本方法采用拉格朗日分解的方式, 亦称为变量复制[39]. 鉴于神经网络的组合结构, 大多数约束仅涉及有限数量的变量. 因此, 该问题可以被分解为定义明确且易于求解的子问题, 为了保证这些子问题解的一致性, 本方法引入了额外的同一性约束. 在原始非凸的原始问题即公式 (5) 的基础上, 将非线性激活等式替换为一个对应其凸包的约束. 由此得到公式 (6) 所示的凸优化问题.

$$\min_{a,\hat{a}} R(X), \text{ where } R(X) = W^{Fa}a^{(m)} + b^F, \text{ s.t. } \begin{cases} \text{cvx\_hull}(a^{(t)}, \hat{a}^{(t)}, l^{(t)}, u^{(t)}),\ t \in [1,m] \\ \hat{a}^{(t)} = W^{aa}a^{(t-1)} + W^{ax}x^{(t)} + b^a,\ t \in [1,m] \\ x^{(t)} \in B_p(x_0^{(t)}, \varepsilon),\ t \in [1,m] \end{cases} \tag{6}$$

公式 (6) 表示的是对激活函数线性近似后得到的优化问题. 由于线性近似是严格的上近似, 该问题求得的任一下界亦是原问题公式 (5) 的一个有效下界. 该问题中 cvx_hull 约束的定义如公式 (4) 所示, 即将约束中原本的 tanh 函数统一用第 3.1 节中定义的近似凸包进行替换. 为了求解该问题, 本文将约束划分为若干约束谓词, 每个谓词所涉及的变量仅出现在网络中相邻的两层. 考虑到 RNN 中第 0 层一般为零向量, 即无“初始状态”, 因而第 1 层变量的值只与该层输入相关. 同时最后一个线性层, 即输出层没有输入, 因而输出层变量的值仅与上一层的状态有关. 除了这两层需要进行特殊处理外, 其他每个约束谓词对应一个网络中的激活层及其后续的线性层. 形式上, 本文为这些约束子集引入了符号简记, 如公式 (7) 所示:

$$\begin{cases} \mathrm{P}(\hat{a}^{(t)},\hat{a}^{(t+1)}) \equiv \begin{cases} \mathrm{cvx\_hull}(a^{(t)},\hat{a}^{(t)},l^{(t)},u^{(t)}) \\ \hat{a}^{(t+1)} = W^{aa}a^{(t)} + W^{ax}x^{(t+1)} + b^{a} \\ x^{(t+1)} \in B_p(x_0^{(t+1)},\varepsilon) \end{cases} \\ \mathrm{P_I}(a^{(0)},\hat{a}^{(1)}) \equiv \begin{cases} a^{(0)} = 0 \\ \hat{a}^{(1)} = W^{aa}a^{(0)} + W^{ax}x^{(1)} + b^{a} \\ x^{(1)} \in B_p(x_0^{(1)},\varepsilon) \end{cases} \\ \mathrm{P_O}(\hat{a}^{(m)},R(X)) \equiv \begin{cases} \mathrm{cvx\_hull}(a^{(m)},\hat{a}^{(m)},l^{(m)},u^{(m)}) \\ R(X) = W^{Fa}a^{(m)} + b^{F} \end{cases} \end{cases} \tag{7}$$

其中, P 表示网络中相邻两层输出构成的约束谓词, $\mathrm{P_I}$ 表示网络第 1 层及其输入构成的约束谓词, $\mathrm{P_O}$ 表示网络输出层与其前一层状态构成的约束谓词. 值得注意的是, 上述谓词的定义中并未显式依赖于第 t 层神经元的激活值 $a^{(t)}$, 因为 $a^{(t)}$ 仅在松弛约束 cvx_hull 内部出现, 并未涉及目标函数. 利用这种约束分组的方式, 可以将公式 (6) 的优化问题简化表示为公式 (8) 的形式.

$$\min_{a,\hat{a}} R(X),\ \text{s.t.} \begin{cases} \mathrm{P_I}(a^{(0)},\hat{a}^{(1)}) \\ \mathrm{P}(\hat{a}^{(t)},\hat{a}^{(t+1)}),\ t \in [1,m-1] \\ \mathrm{P_O}(\hat{a}^{(m)},R(X)) \end{cases} \tag{8}$$

在得到上述原始优化问题关于其约束谓词的重新表述后, 下一步对其进行拉格朗日分解. 具体做法是对激活前的中间层变量进行复制, 将 $\hat{a}^{(t)}$ 变量复制为两份 $\hat{a}_A^{(t)}$ 与 $\hat{a}_B^{(t)}$, 其中 $\hat{a}_A^{(t)}$ 与前一层变量进行组合形成约束子问题而 $\hat{a}_B^{(t)}$ 与后一层变量进行组合. 这使得每个约束谓词都拥有其相关变量的独立副本, 且每一个约束谓词都表示一个可以独立求解的子问题. 在此基础上, 公式 (8) 的优化问题被重写为公式 (9) 的形式.

$$\min_{a,\hat{a}} R(X),\ \text{s.t.} \begin{cases} \mathrm{P_I}(a^{(0)},\hat{a}_A^{(1)}) \\ \mathrm{P}(\hat{a}_B^{(t)},\hat{a}_A^{(t+1)}),\ t \in [1,m-1] \\ \mathrm{P_O}(\hat{a}_B^{(m)},R(X)) \\ \hat{a}_A^{(t)} = \hat{a}_B^{(t)},\ t \in [1,m] \end{cases} \tag{9}$$

注意公式 (8) 与公式 (9) 均为待求解优化问题即公式 (6) 的等价表示. 公式 (9) 除了对变量进行复制外, 还添加了额外的等式约束来保证变量副本之间的同一性. 本方法采用对偶优化对该问题进行求解. 具体而言, 根据对偶优化理论, 引入一组对偶变量 ρ 并将式 (9) 所示的优化问题转换为其对偶问题, 如公式 (10) 所示. 受文献 [34,36] 的启发, 对偶变量 $\rho = [\rho_1,\ldots,\rho_m]$ 被初始化为 $\rho_t = -D^{(t)}(W^{aa})^{\mathrm{T}}D^{(t+1)}\ldots D^{(m)}(W^{Fa})^{\mathrm{T}}$ 对于 $t \in [1,m]$, 其中 W^{aa} 为网络中的权重矩阵参数, 上标 T 表示矩阵转置, $D^{(t)}$ 是一个对角矩阵, $D_{jj}^{(t)}$ 表示在图 4 细粒度近似的 3 种情况下第 t 层第 j 个神经元激活函数的近似上界斜率, 即 $\alpha_{U,j}^{(t)}$.

$$\begin{cases} \max\limits_{\rho} q(\rho),\ \text{where}\ q(\rho) = \min\limits_{a,\hat{a}} \left(R(X) + \sum\limits_{t=1}^{m} \rho_t^{\mathrm{T}} \left(\hat{a}_B^{(t)} - \hat{a}_A^{(t)} \right) \right) \\ \text{s.t.} \begin{cases} \mathrm{P_I}(a^{(0)},\hat{a}_A^{(1)}) \\ \mathrm{P}(\hat{a}_B^{(t)},\hat{a}_A^{(t+1)}),\ t \in [1,m-1] \\ \mathrm{P_O}(\hat{a}_B^{(m)},R(X)) \end{cases} \end{cases} \tag{10}$$

公式 (10) 表示的与公式 (9) 对偶的优化问题, 旨在求出含对偶变量 ρ 的目标函数 $q(\rho)$ 的上界. 根据对偶优化理论中的弱对偶性, 原优化问题的最小值是其对偶问题中最大值的上界, 即任何 ρ 的取值都能给原始问题提供一个有效的下界. 因此, 若公式 (10) 得到求解, 甚至只需求得目标函数 $q(\rho)$ 的某个较大值, 则公式 (9) 直至原问题得到求解. 尽管在实践中将通过优化对偶变量的选择来尽可能获得更紧密的边界, 但可以在优化过程的任意时刻中断迭代计算, 并通过评估 ρ 来获得一个有效的下界. 对于构建好的对偶问题的具体优化求解方法在第 3.3 节中进行详细推导, 包括子问题的求解以及网络整体输出最小值的求解.

3.3 对偶问题的优化求解方法

在使用公式 (10) 的对偶问题替代原优化问题的基础上, 接下来展示如何高效地求解该问题. 具体而言, 本文提出了一种基于超梯度上升的方法, 在给定点 ρ 处, 求解使内层子问题达到最小化的 $\hat{a}_A$ 和 $\hat{a}_B$ 的值, 分别记作 $\hat{a}_A^*$ 和 $\hat{a}_B^*$. 基于确定的 $\hat{a}_A^*$ 和 $\hat{a}_B^*$ 值, 可以计算出超梯度 $\nabla_\rho q = \hat{a}_B^* - \hat{a}_A^*$. 随后, 对 ρ 的更新采用常规的超梯度上升的更新方式, 如公式 (11) 所示:

$$\rho_{\text{new}} = \rho_{\text{old}} + \eta \nabla_\rho q(\rho_{\text{old}}) \tag{11}$$

具体而言, 公式 (11) 给出了求解公式 (10) 所示优化问题的基本步骤, 即利用超梯度对对偶变量 ρ 进行迭代更新从而使目标函数 $q(\rho)$ 的取值增大. 其中, η 表示需要提供的步长, 计算过程也可以使用任意梯度下降的变体, 例如 Adam 优化算法. 通过逐步迭代精化即可求得该问题的解, 即原优化问题的一个下界. 接下来说明如何对子问题中的 $\hat{a}_A^*$ 和 $\hat{a}_B^*$ 进行优化求解. 根据本方法的设计, 每个变量仅涉及一个约束子问题, 因此只需逐个子问题优化线性函数即可. 注意到在 3 种子问题中, $\mathrm{P_I}$ 内的约束只与输入有关, $\mathrm{P_O}$ 仅与前一层变量相关, 而中间层子问题 P 则兼而有之. 因此, 下文首先对子问题 $\mathrm{P_I}$ 和 $\mathrm{P_O}$ 分别进行求解, 综合二者求解的方法即可对子问题 P 进行求解.

3.3.1 $\mathrm{P_I}$ 子问题优化求解

由于 RNN 中第 0 层一般为零向量, 因而第 1 层变量的值仅与该层输入相关. 对受约束于 $\mathrm{P_I}$ 的变量 $x^{(1)}$和$\hat{a}_A^{(1)}$ 进行求解, 等价于求解以下问题, 如公式 (12) 所示:

$$\left[x^{(1)*}, \hat{a}_A^{(1)*}\right] = \underset{x^{(1)}, \hat{a}_A^{(1)}}{\arg\min} -\rho_{\mathrm{I}}^{\mathrm{T}} \hat{a}_A^{(1)}, \text{ s.t. } \hat{a}_A^{(1)} = W^{ax} x^{(1)} + b^a x^{(1)} \in B_p(x_0^{(1)}, \varepsilon) \tag{12}$$

公式 (12) 对公式 (7) 所示的 $\mathrm{P_I}$ 子问题进行求解, 以得到目标参数值 $\hat{a}_A^{(1)*}$ 连同相应输入值 $x^{(1)*}$, 注意 $\mathrm{P_I}$ 与参数 $\hat{a}_B$ 无关. 该问题在输入扰动范围内最小化 $-\rho_{\mathrm{I}}^{\mathrm{T}} W^{ax} x^{(1)}$, 这是一个关于输入的线性函数. 本方法支持输入扰动在多种距离度量范数下的鲁棒半径求解, 包括常见范数 l_1、l_2 以及 l_∞. 具体求解方式如下: 假设 $x^{(1)*} = x_0^{(1)} + D$, 其中 D 为系数矩阵 $-\rho_{\mathrm{I}}^{\mathrm{T}} W^{ax}$ 中每个行向量对应的扰动向量构成的矩阵, 此处加号表示原始输入 $x_0^{(1)}$ 与扰动矩阵 D 进行逐列相加, A 表示系数矩阵 $-\rho_{\mathrm{I}}^{\mathrm{T}} W^{ax}$, 则上述最小化问题可看作最小化 $A \cdot D$ 的对角线元素. 下面针对 3 种距离范数进行求解.

在 l_1 范数扰动下, 为了实现目标, 考虑 A 的行向量 $A_i = [a_{i1}, a_{i2}, \ldots, a_{in}]$, 其对应的扰动向量为 D_i. 找到该行分量绝对值最大的位置: $j_i = \underset{j=1,2,\ldots,n}{\arg\max} |a_{ij}|$, 并对该行扰动向量 D_i 的对应分量设置为 $D_{ij_i} = -\varepsilon \cdot \mathrm{sign}(a_{ij_i})$, 其中 $\mathrm{sign}(x)$ 表示 x 的符号函数, 其余分量取 0, 此时得到第 i 个行向量对应的扰动向量 D_i. 综合 A 中所有行向量的扰动向量 D_i, 获得扰动矩阵 D, 进而最优解如公式 (13) 所示:

$$\begin{cases} x^{(1)*} = x_0^{(1)} + \mathrm{D} \\ \hat{a}_A^{(1)*} = W^{ax} x^{(1)*} + b^a \end{cases} \tag{13}$$

在 l_2 和 l_∞ 范数扰动下, 最优解亦如公式 (13) 所示, 但是 D 的求解过程与 l_1 范数有所不同. 具体而言, 注意到 l_2 范数在扰动空间中表现为球形, 因此对于 A 中每个行向量 $A_i = [a_{i1}, a_{i2}, \ldots, a_{in}]$, 首先求其单位方向向量 $A_{i\_unit} = A_i / \|A_i\|_2$, 随即得到该行对应的扰动向量 $D_i = -\varepsilon \cdot A_{i\_unit}$. 综合 A 中所有行向量的扰动向量 D_i, 获得扰动矩阵 D.

在 l_∞ 范数扰动下, 输入的每个位置的扰动范围不超过 ε. 因此, 对于 A 中每个行向量 $A_i = [a_{i1}, a_{i2}, \ldots, a_{in}]$, 判断其各位置分量的符号, 对该行扰动向量 D_i 的每个对应分量设置为 $D_{ij} = -\varepsilon \cdot \mathrm{sign}(a_{ij})$. 此处与 l_1 范数的不同之处在于 l_1 范数的扰动距离 ε 限制的是输入向量所有分量的扰动之和, 而 l_∞ 范数扰动中输入向量的每个位置分量都有大小为 ε 的扰动自由度. 同理综合所有扰动向量 D_i, 即获得扰动矩阵 D, 进而得到公式 (13) 所示的最优解. 此外, 该范数扰动下的线性规划问题求解还可以简化为: 输入扰动空间由一组上下界 $l^{(1)}$ 和 $u^{(1)}$ 定义, 优化过程根据输入 $x^{(1)}$ 的系数矩阵的符号选择下界 $l^{(1)}$ 或上界 $u^{(1)}$. 具体而言, 对于矩阵 $\rho_{\mathrm{I}}^{\mathrm{T}} W^{ax}$ 中每个行向量大于等于 0 的位置 $x^{(1)}$ 取 $u^{(1)}$, 否则取 $l^{(1)}$. 令 $\mathbb{C}_W$ 表示关于矩阵 W 的选择函数, 最优解可表示为公式 (14) 所示的形式.

$$\begin{cases} x^{(1)*} = \mathbb{C}_{\rho_{\mathrm{I}}^{\mathrm{T}} W^{ax} < 0} \cdot l^{(1)} + \mathbb{C}_{\rho_{\mathrm{I}}^{\mathrm{T}} W^{ax} \geq 0} \cdot u^{(1)} \\ \hat{a}_A^{(1)*} = W^{ax} x^{(1)*} + b^a \end{cases} \tag{14}$$

3.3.2 $\mathrm{P_O}$ 子问题优化求解

由于 RNN 中输出层一般没有输入, 因而输出层变量的值仅与上一层状态相关. 为方便叙述, 将输出层记作第 m+1 层, 即 $\hat{a}_A^{(m+1)} = R(X)$. 对受约束于 $\mathrm{P_O}$ 的变量 $\hat{a}_B^{(m)}$和$R(X)$ 进行求解, 等价于求解如公式 (15) 所示的子问题. 其中 cvx_hull 约束由公式 (4) 给出, 表示对 tanh 激活函数的线性近似.

$$\left[\hat{a}_B^{(m)*}, \hat{a}_A^{(m+1)*}\right] = \underset{\hat{a}_B^{(m)}, \hat{a}_A^{(m+1)}}{\arg\min} \rho_m^{\mathrm{T}} \hat{a}_B^{(m)} - \rho_{m+1}^{\mathrm{T}} \hat{a}_A^{(m+1)}, \text{ s.t. } \begin{cases} \text{cvx\_hull}(a^{(m)}, \hat{a}_B^{(m)}, l^{(m)}, u^{(m)}) \\ \hat{a}_A^{(m+1)} = W^{Fa} a^{(m)} + b^F \end{cases} \tag{15}$$

公式 (15) 对公式 (7) 所示的 $\mathrm{P_O}$ 子问题进行求解, 以得到目标参数值 $\hat{a}_B^{(m)*}$ 以及相应的输出值 $\hat{a}_A^{(m+1)*}$. 利用公式 (15) 中最后一个等式约束进行代换, 可以将目标函数重写为 $\rho_m^{\mathrm{T}} \hat{a}_B^{(m)} - \rho_{m+1}^{\mathrm{T}} W^{Fa} a^{(m)}$. 现在针对图 6 中所示的 3 种不同的激活函数近似情况对该优化问题进行求解.

在图 6(a) 情况下 $u^{(m)} \leqslant 0$, 上界线为左右端点的连线, 下界线为左右端点处 2 条切线共同组成的折线. 如果 $\rho_{m+1}^{\mathrm{T}} W^{Fa}$ 的符号为负, 则优化问题的解对应 $a^{(m)}$ 的下界 $h_L^{(m)}(\hat{a}_B^{(m)})$; 反之如果符号为正则对应 $a^{(m)}$ 的上界 $h_U^{(m)}(\hat{a}_B^{(m)}) = \dfrac{\tanh(u^{(m)}) - \tanh(l^{(m)})}{u^{(m)} - l^{(m)}} \left(\hat{a}_B^{(m)} + \dfrac{\tanh(u^{(m)})(u^{(m)} - l^{(m)})}{\tanh(u^{(m)}) - \tanh(l^{(m)})} - u^{(m)} \right)$. 此时公式 (15) 的子问题可以被重写为公式 (16).

$$\underset{\hat{a}_B^{(m)} \in [l^m, u^m]}{\arg\min} \left(\rho_m^{\mathrm{T}} - [\rho_{m+1}^{\mathrm{T}} W^{Fa}]_+ \frac{\tanh(u^{(m)}) - \tanh(l^{(m)})}{u^{(m)} - l^{(m)}} \right) \hat{a}_B^{(m)} - [\rho_{m+1}^{\mathrm{T}} W^{Fa}]_- h_L^{(m)}(\hat{a}_B^{(m)}) \tag{16}$$

其中, $[A]_-$ 表示矩阵 A 中的负值部分, 定义为 $[A]_- = \min(A, 0)$; 而 $[A]_+$ 表示矩阵 A 中的正值部分, 定义为 $[A]_+ = \max(A, 0)$. 该目标函数及其约束条件完全可以分解到中间层各个对应神经元上, 因此所有问题可以独立求解. 对于每个神经元, 由于折线 $h_L^{(m)}(\hat{a}_B^{(m)})$ 的引入, 问题可被归结为一个下凸的分段线性函数, 这意味着最优解一定位于线段中某个顶点. 可能的顶点包括 $(l_r^{(m)}, \tanh(l_r^{(m)}))$, (i_1, j_1), $(u_r^{(m)}, \tanh(u_r^{(m)}))$. 为了使目标函数达到最小值, 只需在这 3 个点上计算目标函数的值, 并选择其中最小的值对应的点即可. 最后, 令求得的 $\hat{a}_B^{(m)*}$ 经过一次网络中的前向传播即可得到 $\hat{a}_A^{(m+1)*}$.

在图 6(b) 情况下 $l^{(m)} \geqslant 0$, 求解方式与 (a) 类似, 唯一的区别在于上下界线的定义颠倒过来, 下界线为左右端点的连线, 上界线为左右端点处 2 条切线共同组成的折线. 如果 $\rho_{m+1}^{\mathrm{T}} W^{Fa}$ 的符号为负, 则优化问题的解对应 $a^{(m)}$ 的下界 $h_L^{(m)}(\hat{a}_B^{(m)}) = \dfrac{\tanh(u^{(m)}) - \tanh(l^{(m)})}{u^{(m)} - l^{(m)}} \left(\hat{a}_B^{(m)} + \dfrac{\tanh(u^{(m)})(u^{(m)} - l^{(m)})}{\tanh(u^{(m)}) - \tanh(l^{(m)})} - u^{(m)} \right)$; 反之如果符号为正则对应 $a^{(m)}$ 的上界 $h_U^{(m)}(\hat{a}_B^{(m)})$. 此时公式 (15) 所示的子问题可以被重写为公式 (17) 的形式.

$$\underset{\hat{a}_B^{(m)} \in [l^m, u^m]}{\arg\min} \left(\rho_m^{\mathrm{T}} - [\rho_{m+1}^{\mathrm{T}} W^{Fa}]_- \frac{\tanh(u^{(m)}) - \tanh(l^{(m)})}{u^{(m)} - l^{(m)}} \right) \hat{a}_B^{(m)} - [\rho_{m+1}^{\mathrm{T}} W^{Fa}]_+ h_U^{(m)}(\hat{a}_B^{(m)}) \tag{17}$$

其中的符号定义与图 6(a) 情况相同, 可能的最优解顶点包括 $(l_r^{(m)}, \tanh(l_r^{(m)}))$, (i_2, j_2), $(u_r^{(m)}, \tanh(u_r^{(m)}))$. 为了使目标函数最小, 只需在这 3 个点上计算目标函数的值, 并选择其中最小的值对应的点作为 $\hat{a}_B^{(m)*}$ 即可. 同理, 令求得的 $\hat{a}_B^{(m)*}$ 经过一次网络中的前向传播即可得到 $\hat{a}_A^{(m+1)*}$.

在图 6(c) 情况下 $l^{(m)} \leqslant 0 \leqslant u^{(m)}$, 求解较为简单, 因为此时上下边界线均不包含折线. 假设上界线与下界线的斜率分别为 α_u 和 α_l, 截距分别为 β_u 和 β_l. 如果 $\rho_{m+1}^{\mathrm{T}} W^{Fa}$ 的符号为负, 则优化问题的解对应 $a^{(m)}$ 的下界 $h_L^{(m)}(\hat{a}_B^{(m)}) = \alpha_l(\hat{a}_B^{(m)} + \beta_l)$; 反之如果符号为正则对应 $a^{(m)}$ 的上界 $h_U^{(m)}(\hat{a}_B^{(m)}) = \alpha_u(\hat{a}_B^{(m)} + \beta_u)$. 此时公式 (15) 所示的子问题可以被重写为公式 (18) 的形式.

$$\underset{\hat{a}_B^{(m)} \in [l^m, u^m]}{\arg\min} \left(\rho_m^{\mathrm{T}} - [\rho_{m+1}^{\mathrm{T}} W^{Fa}]_- \cdot \alpha_l - [\rho_{m+1}^{\mathrm{T}} W^{Fa}]_+ \cdot \alpha_u \right) \hat{a}_B^{(m)} \tag{18}$$

此优化问题是一个关于 $\hat{a}_B^{(m)}$ 的线性函数, 只需根据系数矩阵的元素符号情况选择 $\hat{a}_B^{(m)}$ 的最大或最小值 $u^{(m)}$、$l^{(m)}$ 即可, 具体的细节与公式 (14) 类似. 同理, 令求得的 $\hat{a}_B^{(m)*}$ 经过一次网络中的前向传播即可得到 $\hat{a}_A^{(m+1)*}$.

3.3.3 P 子问题优化求解

上文对于第 1 层和输出层的子问题求解方式进行了详细的推导, 现在对于更加一般化的中间层子问题 P 进行求解. 根据公式 (7), P 子问题可等价表示为公式 (19) 的形式.

$$\left[\hat{a}_B^{(t)*},\hat{a}_A^{(t+1)*},x^{(t+1)*}\right]=\underset{\hat{a}_B^{(t)},\hat{a}_A^{(t+1)},x^{(t+1)}}{\arg\min}\ \rho_t^{\mathrm{T}}\hat{a}_B^{(t)}-\rho_{t+1}^{\mathrm{T}}\hat{a}_A^{(t+1)},\ \text{s.t.}\begin{cases}\text{cvx\_hull}(a^{(t)},\hat{a}_B^{(t)},l^{(t)},u^{(t)})\\ x^{(t+1)}\in B_p(x_0^{(t+1)},\varepsilon)\\ \hat{a}_A^{(t+1)}=W^{aa}a^{(t)}+W^{ax}x^{(t+1)}+b^a\end{cases}\tag{19}$$

公式 (19) 对公式 (7) 所示的 $\mathrm{P_O}$ 子问题进行求解, 以得到目标参数值 $\hat{a}_B^{(t)*}$ 与 $\hat{a}_A^{(t+1)*}$. 利用其中最后一个等式约束进行代换, 可以将目标函数重写为 $\rho_t^{\mathrm{T}}\hat{a}_B^{(t)}-\rho_{t+1}^{\mathrm{T}}W^{aa}a^{(t)}-\rho_{t+1}^{\mathrm{T}}W^{ax}x^{(t+1)}$. 注意到该目标函数是由两部分线性叠加构成. 前一部分需优化求解中间层变量的值, 求解方式与求解 $\mathrm{P_O}$ 子问题的方式相同. 后一部分则需优化求解输入在扰动空间内的值, 求解方式与求解 $\mathrm{P_I}$ 子问题的方式相同. 因此使用前文所述方式分别优化求解 $\hat{a}_B^{(t)*}$ 与 $x^{(t+1)*}$ 并线性相加即可得到 $\hat{a}_A^{(t+1)*}$ 的值, 此处不再赘述.

3.4 朴素 RNN 鲁棒半径计算方法

第 3.1–3.3 节以近似下界为例详细给出了 RNN 输出层近似边界优化问题的求解方法, 包括拉格朗日分解, 对偶问题的构建以及子问题的优化求解. 近似上界的求解方法与此同理. 除了输出层外, 本方法同样适用于网络中间层变量的近似区间计算: 即更替公式 (5) 中的目标函数为 $\hat{a}^{(t)}$, $t\in[1,m-1]$. 但实际计算中若对每个中间层变量都用公式 (11) 的方法进行对偶变量更新求解, 将导致计算成本过大, 因此在求解公式 (10) 时采用固定 ρ 为初始对偶变量的方式, 下文中称其固定对偶变量法. 同时考虑到更加精确的中间层近似区间将减少线性松弛 (cvx_hull 约束) 所带来的近似误差, 因此实际运算中中间层变量取值范围为固定对偶变量法得到的上下界与基于符号区间传播得到的上下界的交集. 整体而言, 朴素 RNN 的近似区间求解如算法 1 所示. 其中, m 表示网络层数, s 为神经元数目, $R(X)^u$ 和 $R(X)^l$ 分别表示求得的输出层上下界. 对于一个给定的网络模型, 逐层计算中间层神经元的近似上下界 $l^{(t)}$和$u^{(t)}$, 最后得到输出层边界 $R(X)^l$和$R(X)^u$.

算法 1. 朴素 RNN 近似区间求解.

输入: 网络权重与偏置, 扰动距离 ε, 距离度量 p, 原始样本 x_0, 超梯度上升步长 η 及迭代轮次 $step$;
输出: 输出层神经元近似区间的上下界 $R(X)^l$和$R(X)^u$.

1. FOR $t\leftarrow$1 TO $m+1$ DO
2. FOR $r\leftarrow$1 TO s DO
3. IF $t=1$ THEN
4. $l_r^{(t)}$, $u_r^{(t)}\leftarrow$并行求解子问题 $\mathrm{P_I}$ (与单层符号区间传播等价)
5. ELSE IF $1<t\leqslant m$ THEN
6. 利用中间层变量上下界初始化对偶变量 ρ
7. 子问题解 $\hat{a}_B^*$, $\hat{a}_A^*\leftarrow$构建并求解子问题 $\mathrm{P_I}$, P
8. $[l_r^{(t)},u_r^{(t)}]_1\leftarrow$利用固定对偶变量计算公式 (10) 得到当前层下界 $q(\rho)$, 上界同理
9. $[l_r^{(t)},u_r^{(t)}]_2\leftarrow$利用符号区间传播计算当前层上下界
10. 当前层上下界 $[l_r^{(t)},u_r^{(t)}]\leftarrow[l_r^{(t)},u_r^{(t)}]_1\cap[l_r^{(t)},u_r^{(t)}]_2$
11. ELSE $t=m+1$ THEN
12. 利用中间层变量上下界初始化对偶变量 ρ
13. 子问题解 $\hat{a}_B^*$, $\hat{a}_A^*\leftarrow$构建并求解子问题 $\mathrm{P_I}$, P, $\mathrm{P_O}$
14. FOR $step$ DO
15. $\rho_{\text{new}}\leftarrow$利用公式 (11) 计算对偶变量的梯度并更新
16. 输出边界 $R(X)_r^l,R(X)_r^u\leftarrow$利用更新后的对偶变量计算公式 (10) 得到输出下界 $q(\rho_{\text{new}})$, 上界同理
17. RETURN $R(X)^l$, $R(X)^u$

在算法 1 中, 第 3–10 行逐层计算中间层神经元的近似上下界, 计算结果取上述固定对偶变量法与反向区间传

播的交集. 其中, 第 1 层近似边界的计算只需通过对输入子问题进行求解即可得到. 第 11–17 行为输出层上下界的计算过程. 具体而言, 首先对前 m 层的对偶变量 ρ 进行初始化, 其次根据公式 (7) 构建 3 种子问题进而分别利用公式 (12)、(15) 和 (19) 的求解过程实现求解以得到对偶变量的梯度, 最后根据梯度利用公式 (11) 迭代更新对偶变量, 将更新后的对偶变量代入公式 (10) 求得输出层的边界.

在输出层近似区间求解的基础上, 可采用对扰动距离进行二分搜索的方式对网络的鲁棒半径进行求解. 首先, 给定一个扰动距离 ε 作为初始的二分搜索区间上界, 下界默认为 0, 随后调用算法 1 得到输出近似上下界. 之后比较各神经元区间上下界的大小关系确定网络是否鲁棒, 再根据结果调整 ε 的值, 并重新调用算法 1 重新计算, 直至 ε 的二分搜索范围达到预设的精度 acc. 具体而言, 设 $label(X_0)$ 为 j, 则每次计算结束后都会检查 $\forall i \neq j,\ R(X)_j^l > R(X)_i^u$ 是否满足. 若满足则对 ε 进行放大; 反之则对 ε 进行缩小. 综上, 基于二分法的 RNN 鲁棒半径求解过程如算法 2 所示.

算法 2. 基于二分法的 RNN 鲁棒半径求解.

输入: 输入样本真实标签 j, 循环神经网络的权重与偏置参数, 二分区间的初始上界 ε_{ini}, 二分终止精度 acc;
输出: 鲁棒半径 ε_{fin}.

1. $\varepsilon_{\text{ub}} = \varepsilon_{\text{ini}},\ \varepsilon_{\text{lb}} = 0$
2. WHILE $\forall i \neq j,\ R(X)_j^l > R(X)_i^u$ DO
3. $\quad \varepsilon_{lb} \leftarrow \varepsilon_{\text{ub}},\ \varepsilon_{\text{ub}} \leftarrow 2\times\varepsilon_{lb}$
4. $\quad R(X)^l,\ R(X)^u \leftarrow$ 代入 ε_{ub} 调用算法 1 求解得到
5. WHILE $\varepsilon_{\text{ub}} - \varepsilon_{\text{lb}} > acc$ DO
6. $\quad \varepsilon \leftarrow \dfrac{(\varepsilon_{\text{lb}} + \varepsilon_{\text{ub}})}{2}$
7. $\quad R(X)^l,\ R(X)^u \leftarrow$ 代入 ε 调用算法 1 求解得到
8. $\quad$ IF $\forall i \neq j,\ R(X)_j^l > R(X)_i^u$ THEN
9. $\quad\quad \varepsilon_{\text{lb}} \leftarrow \varepsilon$
10. $\quad$ ELSE
11. $\quad\quad \varepsilon_{\text{ub}} \leftarrow \varepsilon$
12. RETURN $\varepsilon_{\text{fin}} \leftarrow \varepsilon_{\text{lb}}$

在算法 2 中, 第 1–4 行通过循环确定二分搜索的区间范围, 即循环结束时, ε_{lb} 表示网络鲁棒半径的下界, 而 ε_{ub} 表示输入在此扰动下网络不具备鲁棒性的一个上界. 第 5–11 行表示鲁棒半径的具体二分搜索过程, 当扰动区间边界之差小于给定精度 acc 后即停止继续搜索, 返回下界 ε_{lb} 作为本方法最终求得的鲁棒半径 ε_{fin}.

4 LSTM 鲁棒半径求解

本节介绍 LSTM 模型鲁棒半径求解方法的具体流程和细节. 回顾第 2 节对门控循环神经网络 LSTM 的内部结构与计算流程进行了介绍, 图例和每一层的计算过程分别如图 2 和公式 (2) 所示. 特殊点在于 LSTM 的隐藏层中包含两类中间变量 $a^{(t)}$ 与 $c^{(t)}$, 且层与层之间的传播过程涉及更多的非线性计算. 然而, 其每层的记忆单元整体仍然符合神经网络的层次化结构, 即网络中的每个约束只作用在相邻的两层神经元及其输入上, 因此本文提出的拉格朗日分解结合对偶优化的总体方法仍然适用于 LSTM 鲁棒半径的求解. 考虑到朴素 RNN 中隐藏层变量之间的约束仅有一层激活函数与线性变换, 而 LSTM 中除多种激活函数外还存在逻辑门耦合带来的二元非线性函数, 对此本方法采用限界平面整体近似的方式来构建各层记忆单元之间的 cvx_hull 约束. 此外, 由于 LSTM 还维护一个记忆信息 $c^{(t)}$, 因此在拉格朗日分解时除了需要复制变量 $a^{(t)}$, 对 $c^{(t)}$ 也需同样处理. 本节首先提出基于采样与线性规划 (linear programming, LP) 的二元非线性限界平面计算方法, 为后续的优化问题求解打下基础. 其次, 形式化地表述 LSTM 输出层神经元区间上下界的优化问题, 并对其进行拉格朗日分解, 得到一组相邻层之间可以独立求解的子问题且子问题之间通过同一性约束进行关联, 以保证整体解的一致性. 最后, 描述子问题的求解过程及对偶变量

的优化方法, 并给出完整的 LSTM 输出边界求解算法.

4.1 非线性限界平面计算与 cvx_hull 约束的构建

LSTM 每一层的前向传播过程已在公式 (2) 中进行了描述, 此处定义 LSTM 的遗忘门、输入门、输出门、候选门运算结果的激活前值如公式 (20) 所示, 且对应区间上下界可以通过反向回溯迭代计算得到.

$$\begin{cases} y^{f(t)} = W^{fx}x^{(t)} + W^{fa}a^{(t-1)} + b^{f} \\ y^{g(t)} = W^{gx}x^{(t)} + W^{ga}a^{(t-1)} + b^{g} \\ y^{i(t)} = W^{ix}x^{(t)} + W^{ia}a^{(t-1)} + b^{i} \\ y^{o(t)} = W^{ox}x^{(t)} + W^{oa}a^{(t-1)} + b^{o} \end{cases} \tag{20}$$

公式 (20) 给出了 LSTM 中 4 种门结构激活前值的基本表示, 后续会用其定义 cvx_hull 约束. 对 LSTM 内部结构的分析可以观察到, LSTM 在计算隐藏层状态与记忆信息的过程中除了矩阵运算之外, 还有两类非线性函数的乘积项: $\sigma(v)z$ 与 $\sigma(v)\tanh(z)$, 在给定输入区间后的图像为一个曲面. 显然, 无法利用朴素 RNN 中单一激活函数的近似方法对该乘积项进行近似. 因此, 本方法对近似方式进行扩展, 利用二元线性函数 (限界平面) 对其进行整体近似. 公式 (21) 给出了上述二元非线性函数上、下两个限界平面的基本函数表示, 其中, 上、下限界平面在定义域内必须严格高于或者低于待近似的曲面.

$$\begin{cases} h_{U,r}^{cross(t)}(v,z) = \alpha_{U,r}^{cross(t)}v + \beta_{U,r}^{cross(t)}z + \gamma_{U,r}^{cross(t)} \\ h_{L,r}^{cross(t)}(v,z) = \alpha_{L,r}^{cross(t)}v + \beta_{L,r}^{cross(t)}z + \gamma_{L,r}^{cross(t)} \\ h_{L,r}^{cross(t)}(v,z) \leqslant f(v,z) \leqslant h_{U,r}^{cross(t)}(v,z) \end{cases} \tag{21}$$

公式 (21) 中 $h_{U,r}^{cross(t)}(v,z)$ 代表上界平面函数而 $h_{L,r}^{cross(t)}(v,z)$ 表示下界平面. 注意到平面参数均有上标 $cross$, 表示 $f(v,z)$ 乘积项产生的位置. 在一个记忆单元中分别有 3 种非线性耦合, 分别是输入门与候选门、输出门与当前层记忆信息及遗忘门与上一层记忆信息, 简单表示为 $cross \in \{ig, oc, fc\}$. 类比一元激活函数近似的过程, 上下限界平面的参数由输入变量 v 和 z 的取值范围共同决定, 限界平面与曲面的示意图如图 7(a) 所示. 为了计算出紧密的限界平面方程的参数, 本文用平面与待近似曲面之间的空间距离作为近似程度的指标. 因此, 最小化二者之间的距离即可得到理想的限界平面, 该目标可以转化为问题 (22).

$$\begin{cases} \min\limits_{\alpha_L,\beta_L,\gamma_L} \int_{(v,z)\in S} (f(v,z) - (\alpha_L v + \beta_L z + \gamma_L)), \text{ s.t. } \alpha_L v + \beta_L z + \gamma_L \leqslant f(v,z),\ \forall (v,z) \in S \\ \min\limits_{\alpha_U,\beta_U,\gamma_U} \int_{(v,z)\in S} ((\alpha_U v + \beta_U z + \gamma_U) - f(v,z)), \text{ s.t. } \alpha_U v + \beta_U z + \gamma_U \geqslant f(v,z),\ \forall (v,z) \in S \end{cases} \tag{22}$$

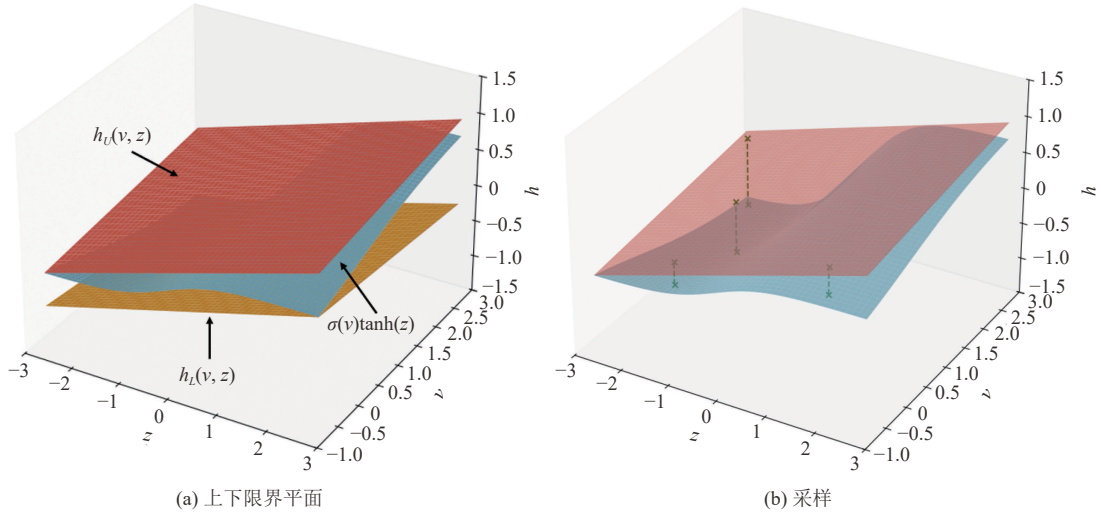

图 7 待近似曲面的限界平面与采样示意图

公式 (22) 采用优化问题的形式表述了二元非线性函数的线性近似问题, 即上下限界平面的选取需尽可能接近待近似曲面, 这是 LSTM 鲁棒半径求解所需处理的关键问题. 其中, $f(v,z)$ 代表 $\sigma(v)z$ 或 $\sigma(v)\tanh(z)$, $S=[l_v,u_v]\times[l_z,u_z]$. 为了解决该问题, 本文将平面的确定过程分为两部分: 首先基于输入采样与 LP 确定一个初始的近似平面, 该平面可以确定在所有的采样点中均满足曲面在上下限界平面之间, 并且让二者之间的空间距离最小. 其次, 由于采样点数量的限制, 初始平面可能存在误差导致未能完全包含待近似曲面, 因此再通过调整平面偏移量参数 γ 使其严格满足约束, 接下来对上述过程进行分别介绍.

4.1.1 基于采样与 LP 确定初始平面

在输入域 S 内进行选点采样. 设采样总数为 N, 采样时在区间 $[l_v,u_v]$ 与 $[l_z,u_z]$ 内分别选择 N 个值, 对其进行依次组合即可获得输入域中的采样坐标. 设采样坐标集合为 $D=\{(v_1,z_1),\ldots,(v_N,z_N)\}$. 在求解下限界平面时问题 (22) 可以离散化为公式 (23) 所示的优化问题, 上限界平面同理.

$$\min_{\alpha_L,\beta_L,\gamma_L}\sum_{i=1}^{N}(f(v_i,z_i)-(\alpha_L v_i+\beta_L z_i+\gamma_L)),\ \text{s.t.}\ \cap_{i=1}^{N}\alpha_L v_i+\beta_L z_i+\gamma_L\leqslant f(v_i,z_i) \tag{23}$$

在公式 (23) 中, 目标函数为所有采样点与对应近似平面之上的投影点距离之和, 越小的距离也代表着越小的空间体积, 约束集合代表曲面中每个采样点都高于下界平面. 该问题在采样点数量较低时可以使用主流的 LP 求解器例如 Gurobi 进行求解. 图 7(b) 为曲面采样点与平面投影之间的距离示意图, 采样点与投影点用×号表示, 二者之间的距离在图中表示为红色虚线.

4.1.2 平面偏移量调整

求解优化问题 (23) 得到的平面仅保证了原始曲面的采样部分被正确近似, 其他部分也许会低于近似平面. 为了整体近似足够精确, 本文对初始平面方程 $\alpha_L v+\beta_L z+\gamma_L$ 中的偏移量参数 γ_L 进行调整. 首先计算曲面与初始平面之间距离的最小值 $\Delta_L=\min_{(v,z)\in S}f(v,z)-(\alpha_L v+\beta_L z+\gamma_L)$, 然后更新 γ_L 为 $\gamma_L+\Delta_L$, 调整后即可保障近似平面的精确性. 接下来分别对两类交叉项近似平面 Δ_L 进行求解.

对于 $f(v,z)=\sigma(v)\tanh(z)$, 定义最小化函数 $G(v,z)$ 并计算其偏导来获得其极点, 进而获得定义域内的最小值. 公式 (24) 给出了该情形下的最小化函数及其偏导.

$$\begin{cases} G(v,z)=\sigma(v)\tanh(z)-(\alpha_L v+\beta_L z+\gamma_L)\\ \dfrac{\partial G}{\partial v}=\sigma(v)\tanh(z)(1-\sigma(v))-\alpha_L\\ \dfrac{\partial G}{\partial z}=\sigma(v)(1-\tanh^2(z))-\beta_L \end{cases} \tag{24}$$

根据极值点求解规则, 公式 (25) 中偏导方程的根以及定义域 S 内的 4 个端点为可能的极值点. 分别计算最小化函数在这些点的值, 即可确定最小值 Δ_L, 并随之更新 γ_L. 调整后新的下限界平面能够可靠地对曲面进行近似.

对于交叉项 $f(v,z)=\sigma(v)z$, 公式 (25) 给出了该情形下的二元函数 $G(v,z)$ 与其偏导. 偏移量的调整过程与上一情形同理, 此处不再赘述.

$$\begin{cases} G(v,z)=\sigma(v)z-(\alpha_L v+\beta_L z+\gamma_L)\\ \dfrac{\partial G}{\partial v}=\sigma(v)z(1-\sigma(v))-\alpha_L\\ \dfrac{\partial G}{\partial z}=\sigma(v)-\beta_L \end{cases} \tag{25}$$

4.1.3 定义近似约束 cvx_hull

在获得交叉项的上下界平面近似后, 即可构建相邻两层神经元之间约束的凸包近似. 公式 (26) 给出了 LSTM 结构的凸包近似约束 cvx_hull, 定义了第 t 层输出 $a^{(t)}$ 和记忆 $c^{(t)}$ 取值的近似上下界, 这些边界与第 t 层输入 $x^{(t)}$ 以及前一层的输出 $a^{(t-1)}$ 和记忆 $c^{(t-1)}$ 有关.

$$\text{cvx\_hull}(x^{(t)},a^{(t)},c^{(t)},a^{(t-1)},c^{(t-1)})\equiv\begin{cases} h_{L,r}^{oc(t)}(y^{o(t)},c^{(t)})\leqslant a_r^{(t)}\leqslant h_{U,r}^{oc(t)}(y^{o(t)},c^{(t)})\\ h_{L,r}^{fc(t)}(y^{f(t)},c^{(t-1)})+h_{L,r}^{ig(t)}(y^{i(t)},y^{g(t)})\leqslant c_r^{(t)}\leqslant h_{U,r}^{fc(t)}(y^{f(t)},c^{(t-1)})+h_{U,r}^{ig(t)}(y^{i(t)},y^{g(t)}) \end{cases} \tag{26}$$

公式 (26) 中, 近似平面 h 的定义如公式 (21) 所示, y 的定义如公式 (20) 所示, 表示每个逻辑门结构的激活前值. 通过限界平面将 $a^{(t)}$ 和 $c^{(t)}$ 涉及的交叉项约束进行近似替换后, 即可通过权重矩阵反向区间传播的方式得到 $a^{(t)}$ 和 $c^{(t)}$ 关于 $a^{(t-1)}$ 和 $c^{(t-1)}$ 与当前层输入 $x^{(t)}$ 的线性表达式, 便于后续对分解之后的子优化问题进行求解.

4.2 LSTM 对偶问题的构建与求解

在求解 LSTM 鲁棒性验证问题时, 同样可以将鲁棒性验证问题描述为网络输出最大最小值的约束求解问题. 对于一个 m 层的 LSTM 网络, 输出表示为 $a^{(m)}$, 同时将每一层的记忆单元用公式 (26) 所定义的 cvx_hull 约束进行近似, 从而得到公式 (27) 所示的优化问题. 此即为待解决的 LSTM 鲁棒性验证问题, 本节采用对偶优化技术对其进行转换和求解.

$$\min_{a,c} a^{(\mathrm{m})},\ \text{s.t.}\ \begin{cases} \text{cvx\_hull}(x^{(t)},a^{(t)},c^{(t)},a^{(t-1)},c^{(t-1)}),\ t\in[1,m] \\ x^{(t)}\in B_p(x_0^{(t)},\varepsilon),\ t\in[1,m] \end{cases} \tag{27}$$

注意到公式 (27) 中 cvx_hull 约束对每层的记忆单元做了整体近似, 因此中间层变量没有朴素 RNN 中的激活前后之分. 同理, 为了实现问题分解, 将约束划分为若干约束谓词, 每个谓词中涉及的变量仅出现在网络相邻的两层中, 即对应网络中的一层记忆单元. 利用约束分组可以将公式 (27) 的优化问题等价转换为公式 (28) 的形式.

$$\begin{cases} \min\limits_{a,c} a^{(m)},\ \text{s.t. P}(x^{(t)},a^{(t)},c^{(t)},a^{(t-1)},c^{(t-1)}),\ t\in[1,m] \\ \text{P}(x^{(t)},a^{(t)},c^{(t)},a^{(t-1)},c^{(t-1)}) \equiv \begin{cases} \text{cvx\_hull}(\text{x}^{(t)},a^{(t)},c^{(t)},a^{(t-1)},c^{(t-1)}) \\ x^{(t)}\in B_p(x_0^{(t)},\varepsilon) \end{cases} \end{cases} \tag{28}$$

公式 (28) 的优化问题对公式 (27) 中的约束进行了整理, 将第 t 层的相关约束记为约束谓词 $\text{P}(x^{(t)},a^{(t)},c^{(t)},a^{(t-1)},c^{(t-1)})$, 简称为子问题 P. 随后, 对该问题构造拉格朗日分解. 在此构造中, 对中间层变量进行复制的过程与朴素 RNN 中的处理有一点细节不同: 除了将 $a^{(t)}$ 变量复制为 $a_A^{(t)}$ 与 $a_B^{(t)}$, 还需要将 $c^{(t)}$ 变量复制为 $c_A^{(t)}$ 与 $c_B^{(t)}$. 其中, $a_A^{(t)}$和$c_A^{(t)}$ 与前一层变量进行组合, 而 $a_B^{(t)}$和$c_B^{(t)}$ 与后一层变量组合, 使得每个约束谓词都表示一个可以独立求解的子问题. 在此基础上将公式 (28) 重写为公式 (29) 所示的优化问题, 其为公式 (28) 进行变量复制处理后的等价形式.

$$\min_{a,c} a_A^{(\mathrm{m})},\ \text{s.t.}\ \begin{cases} \text{P}(x^{(t)},a_A^{(t)},c_A^{(t)},a_B^{(t-1)},c_B^{(t-1)}),\ t\in[1,m] \\ a_A^{(t)}=a_B^{(t)},\ t\in[1,m] \\ c_A^{(t)}=c_B^{(t)},\ t\in[1,m] \end{cases} \tag{29}$$

注意到公式 (29) 中的同一性约束同时考虑了两种隐藏层变量, 类似地引入对偶变量 ρ、λ 构建出其对偶优化问题如公式 (30) 所示. 上述两种对偶变量的初始化规则与朴素 RNN 中的类似, 但区别在于此处同一层记忆单元权重矩阵数量更多, 在二元曲面近似时变量之间的关系更为复杂, 此处给出第 $m-1$ 层对偶变量的初始化方法: $\rho_{m-1}=-W_{oa}^{\mathrm{T}}\alpha_U^{oc(m-1)}-(W_{fa}^{\mathrm{T}}\alpha_U^{fc(m-1)}+W_{ia}^{\mathrm{T}}\alpha_U^{ig(m-1)}+W_{ga}^{\mathrm{T}}\beta_U^{ig(m-1)})\beta_U^{oc(m-1)}$, $\lambda_{m-1}=-\beta_U^{fc(m-1)}\beta_U^{oc(m-1)}$. 其中 α和β 均为对角矩阵, 对角线上每个元素表示当前层对应神经元二元非线性项近似平面系数, 上标表示层数以及非线性项类型, 下标 U 表示平面上界.

$$\begin{cases} \max\limits_{\rho,\lambda} q(\rho,\lambda),\ \text{where } q(\rho,\lambda)=\min\limits_{a,c} a_A^{(m)}+\sum\limits_{t=1}^{m}\rho_t^{\mathrm{T}}(a_B^{(t)}-a_A^{(t)})+\sum\limits_{t=1}^{m}\lambda_t^{\mathrm{T}}(c_B^{(t)}-c_A^{(t)}) \\ \text{s.t. P}(x^{(t)},a_A^{(t)},c_A^{(t)},\text{a}_B^{(t-1)},c_B^{(t-1)}),\ t\in[1,m] \end{cases} \tag{30}$$

公式 (30) 表示与公式 (29) 对偶的优化问题, 旨在求出含对偶变量 ρ、λ 的目标函数 $q(\rho,\lambda)$ 的上界. 根据对偶优化理论中的弱对偶性, 原优化问题的最小值是其对偶问题中最大值的上界, 即对偶变量的任何取值都能给原始问题提供一个有效的下界. 因此, 若公式 (30) 得到求解, 甚至只需求得目标函数 $q(\rho,\lambda)$ 的某个较大值, 则公式 (29) 直至原问题 (27) 得到求解.

对于构建好的对偶问题的具体优化求解方法类似于朴素 RNN, 使用基于超梯度上升的方法. 具体而言, 对于公式 (30) 所示的优化问题, 在给定点 ρ λ 处, 获取其超梯度需要计算使内层子问题达到最小化的 a_A、a_B、c_A、c_B 的值, 分别记作 a_A^*、a_B^*、c_A^*、c_B^*. 基于它们可以计算出超梯度 $\nabla_\rho q=a_B^*-a_A^*$, $\nabla_\lambda q=c_B^*-c_A^*$. 随后, 对 ρ、λ 的更新采

用超梯度上升更新方式, 如公式 (31) 所示:

$$\begin{cases} \rho_{\text{new}} = \rho_{\text{old}} + \eta_\rho \nabla_\rho q(\rho_{\text{old}}) \\ \lambda_{\text{new}} = \lambda_{\text{old}} + \eta_\lambda \nabla_\lambda q(\lambda_{\text{old}}) \end{cases} \tag{31}$$

具体而言, 公式 (31) 给出了求解公式 (30) 所示优化问题的基本步骤, 即利用超梯度对两种对偶变量进行迭代更新从而使目标函数 $q(\rho,\lambda)$ 的取值增大. 其中, $\eta=[\eta_\rho,\eta_\lambda]$ 表示需要提供的步长. 通过逐步迭代精化即可求得该问题的解, 即原优化问题的一个下界.

最后说明如何对子问题 P 进行求解. 公式 (32) 根据公式 (28) 中子问题 P 的定义对其进行求解, 以得到目标参数值 a_A^*、a_B^*、c_A^*、c_B^*.

$$\begin{cases} \left\lfloor c_B^{(t-1)*}, a_B^{(t-1)*}, c_A^{(t)*}, a_A^{(t)*}, x^{(t)*} \right\rfloor = \underset{c_B^{(t-1)}, a_B^{(t-1)}, c_A^{(t)}, a_A^{(t)}, x^{(t)}}{\arg\min} \rho_{t-1}^{\mathrm{T}} a_B^{(t-1)} - \rho_t^{\mathrm{T}} a_A^{(t)} + \lambda_{t-1}^{\mathrm{T}} c_B^{(t-1)} - \lambda_t^T c_A^{(t)} \\ \text{s.t.} \begin{cases} \text{cvx\_hull}(x^{(t)}, a^{(t)}, c^{(t)}, a^{(t-1)}, c^{(t-1)}) \\ x^{(t)} \in B_p(x_0^{(t)}, \varepsilon) \end{cases} \end{cases} \tag{32}$$

由于在 cvx_hull 约束的定义中对于二元非线性曲面做了限界平面近似, 因此公式 (32) 中 $a_A^{(t)}$ 可以根据系数 $-\rho_t^{\mathrm{T}}$ 的正负情况选择对应的限界平面进行近似. 具体而言, 将目标函数中的 $a_A^{(t)}$ 替换为关于 $c_A^{(t)}$ 与 $y^{o(t)}$ 的表达式, 令其与 $-\lambda_t^{\mathrm{T}} c_A^{(t)}$ 合并系数后再次根据系数的符号进行限界平面选择, 得到关于 $y^{f(t)}$、$y^{i(t)}$、$y^{g(t)}$、$c_B^{(t-1)}$ 的线性表达式. 根据公式 (20) 可知 $y^{(t)}$ 本身与当前层输入 $x^{(t)}$, 上一层状态 $a_B^{(t-1)}$ 呈线性关系, 因此目标表达式可以被看作一个关于变量 $a_B^{(t-1)}$、$x^{(t)}$、$c_B^{(t-1)}$ 的线性表达式, 根据系数矩阵的符号即可在 $a_B^{(t-1)}$、$c_B^{(t-1)}$ 的上下界范围内选择相应的最大或最小值, 从而得到 a_B^*、c_B^*. 对于输入 $x^{(t)*}$, 可以利用和朴素 RNN 子问题求解中 $\mathrm{P_I}$ 相同的求解方式, 根据系数矩阵的符号和扰动的范数类型来确定, 至此可以根据得到的 a_B^*、c_B^* 以及 $x^{(t)*}$ 的值进行一次前向传播, 从而得到 a_A^*、c_A^*. 以上即为子问题 P 的求解过程.

4.3 LSTM 鲁棒半径计算方法

根据上文描述, 对单个 LSTM 记忆单元进行求解需要进行 3 次平面近似, 而在计算限界平面参数之前需要得到相关门结构的近似上下界. 考虑到曲面近似以及记忆单元中两种对偶变量的计算成本, 本方法不太适合用来求解中间层变量的上下界, 这一点与朴素 RNN 不同. 实践中在计算 LSTM 的中间层变量上下界时使用类似于 POPQORN[24]的回溯迭代求解方法, 值得注意的是由于本方法采用了和 POPQORN 中不同的基于采样与线性优化的二元限界平面求解方法, 因此其效率和精度也与之不同. 综合上述分析, LSTM 输出层近似区间求解方法如算法 3 所示.

算法 3. LSTM 近似区间求解.

输入: LSTM 网络权重与偏置, 扰动范围 ε, 距离度量 p, 原始样本 x_0, 更新步长 η_ρ、η_λ 及迭代轮次 $step$;
输出: 输出层神经元近似区间的上下界.

1. FOR $t \leftarrow 1$ TO m DO
2. FOR $r \leftarrow 1$ TO s DO
3. $l_r^{i(t)}, l_r^{f(t)}, l_r^{g(t)}, l_r^{o(t)}, u_r^{i(t)}, u_r^{f(t)}, u_r^{g(t)}, u_r^{o(t)} \leftarrow$使用回溯迭代求解计算
4. $h_{L,r}^{fc(t)}(y_r^{f(t)}, c_r^{(t-1)}) \leftarrow \text{2D\_BOUND}(l_r^{f(t)}, u_r^{f(t)}, l_r^{c(t-1)}, u_r^{c(t-1)}, fc, low)$
5. $h_{U,r}^{fc(t)}(y_r^{f(t)}, c_r^{(t-1)}) \leftarrow \text{2D\_BOUND}(l_r^{f(t)}, u_r^{f(t)}, l_r^{c(t-1)}, u_r^{c(t-1)}, fc, up)$
6. $h_{L,r}^{ig(t)}(y_r^{i(t)}, y_r^{g(t)}) \leftarrow \text{2D\_BOUND}(l_r^{i(t)}, u_r^{i(t)}, l_r^{g(t)}, u_r^{g(t)}, ig, low)$
7. $h_{U,r}^{ig(t)}(y_r^{i(t)}, y_r^{g(t)}) \leftarrow \text{2D\_BOUND}(l_r^{i(t)}, u_r^{i(t)}, l_r^{g(t)}, u_r^{g(t)}, ig, up)$
8. $l_r^{c(t)}, u_r^{c(t)} \leftarrow$使用回溯迭代计算
9. $h_{L,r}^{oc(t)}(y_r^{o(t)}, c_r^{(t)}) \leftarrow \text{2D\_BOUND}(l_r^{o(t)}, u_r^{o(t)}, l_r^{c(t)}, u_r^{c(t)}, oc, low)$
10. $h_{U,r}^{oc(t)}(y_r^{o(t)}, c_r^{(t)}) \leftarrow \text{2D\_BOUND}(l_r^{o(t)}, u_r^{o(t)}, l_r^{c(t)}, u_r^{c(t)}, oc, up)$

11. 利用中间层变量上下界及近似平面系数初始化对偶变量ρ、λ
12. 子问题解 a_A^*, a_B^*, c_A^*, c_B^* ←构建并求解子问题 P
13. FOR i←1 TO $step$ DO
14. ρ_{new}, λ_{new} ←利用公式 (31) 计算对偶变量的梯度并更新
15. 输出边界 $l^{a(m)}, u^{a(m)}$ ←利用更新后的对偶变量计算公式 (30) 得到输出下界 $q(\rho_{\text{new}}, \lambda_{\text{new}})$, 上界同理
16. RETURN $l^{a(m)}, u^{a(m)}$

在算法 3 中, m 和 s 仍分别表示网络的层数和神经元数目. 第 1–10 行是中间层变量上下界计算以及近似平面系数的确定过程, 相关记号的定义在公式 (20) 和 (21) 中给出. 第 4.1 节提出的平面系数计算过程在算法中表述为 2D_BOUND 函数, 利用公式 (23)–(25) 根据相乘变量的类型以及对应的近似区间求得可靠的限界平面. 第 11、12 行根据式 (32) 及其求解过程实现对偶变量的初始化以及内层子问题的求解, 得到关键参数值进而求得对偶变量的梯度. 第 13、14 行根据梯度利用公式 (31) 为对偶变量执行迭代更新过程. 最终, 第 15、16 行将更新后的对偶变量代入公式 (30) 得到输出层的近似区间上下界. 在此基础上, 同样基于二分法, 将算法 2 中对算法 1 的调用替换为调用算法 3, 即可求得 LSTM 模型的鲁棒半径.

5 实验分析

本节基于上文提出的朴素 RNN 与 LSTM 模型的鲁棒半径计算方法, 实现了一套 RNN 验证工具, 命名为 VRNN-LD, 同时通过实验对 VRNN-LD 进行评估, 并与已有同类验证方法 POPQORN 进行对比分析. 同为 RNN 验证的 CERT-RNN 方法仅支持 l_∞ 范数下的鲁棒半径求解, 不适合进行各种距离度量下的全面比较. 工具整体基于 Python 3.8 实现, 并基于 PyTorch 框架来训练 RNN 模型与并行计算. LP 求解器使用 Gurobi 10.2 且实验均在 64 位 Ubuntu 22.04 LTS 平台上进行, CPU 为 20 核的 Intel Core i7-14700, 内存为 16 GB.

实验中采用两种数据集进行循环神经网络的训练与验证, 分别为 MNIST 手写数字分类数据集以及 UIUC 提出的 CogCompQC 问题分类数据集[40]. MNIST 数据集为 0–9 的灰度手写数字图像, 每张图像的规格为 28×28, 在训练时统一将样本进行均值为 0, 标准差为 1 的归一化. 在 VNN-COMP 中使用到的其他数据集由于输入维度过低、面向前馈和卷积网络等原因在本次实验中未被考虑. 部分数据样本示例如图 8 所示. 实验为了匹配网络的层数, 会将原始输入进行切分, 例如对于一个 7 层的 RNN 模型, 其输入样本格式会被调整为 7×112, 即每一层的网络输入为 112 个像素.

图 8 MNIST 数据集示例

实际应用中 RNN 常被用于序列信息识别与自然语言处理, 因此实验引入基于文本类数据集的网络模型. CogCompQC 数据集主要用于训练文本问题分类网络, 表现为根据每个问题的文本内容, 将其归类到预定义的类别中. 具体而言, 该数据集中的问题均来源于现实场景, 其输出类别按照粗粒度共分为 6 类: ABBR (关于缩写或定义)、ENTY (关于实体类)、DESC (关于描述)、HUM (关于人)、LOC (关于地理位置) 以及 NUM (关于数值), 其部分样本示例如图 9 所示. 文本数据作为输入时, 每个单词作为 RNN 中一个时间步上的输入. 在预定长度的网络结构中, 处理较短的句子时在后续的输入中补充零向量, 而较长的句子中超过网络层数的部分则会被截断. 实验中使用预训练的词向量将每个单词转换成一个固定长度 100 的向量, 形成输入矩阵.

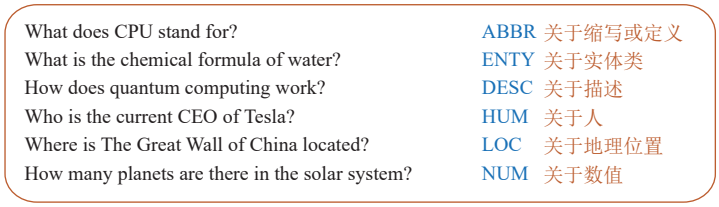

图 9 CogCompQC 数据集示例

5.1 朴素 RNN 鲁棒半径计算实验与分析

为了分析对比 VRNN-LD 与 POPQORN 的求解性能, 本文在多个具有不同层数与隐藏层尺寸的朴素 RNN 模型上面向多种扰动距离进行实验. 表 1 给出了两种方法在 6 类不同的 MNIST 模型与 3 种不同的范数扰动下求得的鲁棒半径, 从左到右分别为最大值、最小值以及平均值. 网络模型中第 1 个指标为隐藏层的规模, 第 2 个指标为模型的层数. 例如 RNN×64×7 模型中隐藏层神经元为 64 而层数为 7, 下同. 6 种模型层数为 4–14 不等, 隐藏层神经元数目为 32 或 64. 求解过程中对偶变量的更新步长 η 随着优化的次数而均匀下降, 初始化为 0.1 而最小为 0.01. 同时鲁棒半径求解的输入数据采用 MNIST 测试集中被模型正确分类的 200 个样本.

表 1 MNIST 数据集鲁棒半径对比

网络模型	距离度量	POPQORN (最大值 | 最小值 | 平均值)	VRNN-LD (最大值 | 最小值 | 平均值)	均值提升率 (%)
RNN×32×4	l_1	1.6344 | 0.2656 | 0.8690	2.0118 | 0.2917 | 1.0228	17.7
	l_2	0.3434 | 0.0552 | 0.1814	0.3748 | 0.0716 | 0.2150	18.5
	l_∞	0.0305 | 0.0049 | 0.0161	0.0377 | 0.0058 | 0.0189	17.9
RNN×64×4	l_1	2.0422 | 0.5566 | 1.0926	2.4353 | 0.6157 | 1.3057	19.5
	l_2	0.4016 | 0.1132 | 0.2209	0.4681 | 0.1320 | 0.2682	21.4
	l_∞	0.0352 | 0.0100 | 0.0194	0.0438 | 0.0124 | 0.0235	21.0
RNN×32×7	l_1	0.5813 | 0.0161 | 0.3348	0.7051 | 0.0183 | 0.4141	23.7
	l_2	0.1490 | 0.0041 | 0.0905	0.1941 | 0.0046 | 0.1112	22.9
	l_∞	0.0177 | 0.0005 | 0.0108	0.0220 | 0.0008 | 0.0133	23.4
RNN×64×7	l_1	0.7648 | 0.2375 | 0.4493	0.8587 | 0.2866 | 0.5486	22.1
	l_2	0.1998 | 0.0615 | 0.1167	0.2230 | 0.0737 | 0.1442	23.6
	l_∞	0.0234 | 0.0073 | 0.0137	0.0294 | 0.0092 | 0.0171	25.0
RNN×32×14	l_1	0.2744 | 0.0168 | 0.1390	0.3527 | 0.0255 | 0.1768	27.2
	l_2	0.0935 | 0.0057 | 0.0484	0.1308 | 0.0116 | 0.0620	28.0
	l_∞	0.0157 | 0.0010 | 0.0082	0.0229 | 0.0025 | 0.0106	29.4
RNN×64×14	l_1	0.2506 | 0.0012 | 0.1522	0.3188 | 0.0018 | 0.1945	27.8
	l_2	0.0840 | 0.0004 | 0.0510	0.1274 | 0.0005 | 0.0650	27.5
	l_∞	0.0139 | 0.0001 | 0.0085	0.0193 | 0.0001 | 0.0108	27.3

表 1 中的结果表明, 在所有 MNIST 网络模型以及不同的距离度量下, VRNN-LD 求解出的鲁棒半径均值对比 POPQORN 提升了 17.7%–29.4% 不等, 同时鲁棒半径的最大值和最小值也有显著提升, 这说明了 VRNN-LD 较已有方法具备更优的求解性能. 同时, 表 1 也展示出 VRNN-LD 相较于 POPQORN 在模型层数较深时有更高的鲁棒半径均值提升率. 具体而言, 对于 4 层的网络提升率为 17.7%–21.4%, 对于 7 层的网络提升率为 22.1%–25.0%, 而对于 14 层的网络提升率为 27.2%–29.4%. 这是由于边界计算是一个逐层迭代的过程, 在计算过程中网络中间层的变量上下界首先被作为求解目标, 将固定对偶变量法与回溯迭代法得到的结果交集作为中间变量范围, 在输出变量的区间求解中更精确的中间变量范围会进一步降低近似误差. 这一结果也说明本方法的精度优势对于较深的 RNN 模型更为明显. 另一方面, 由于中间层变量区间求解的过程同时利用了反向区间传播与固定对偶变量法, 其

求解时间成本要高于 POPQORN.

对于 CogCompQC 问题分类数据集, 其问题样本长度普遍在 5–15 个单词之间. 表 2 给出了两种方法在层数为 16 的两种模型上求得的鲁棒半径, 其中隐藏层神经元数目分别为 32 和 64, 输入数据为 100 个被网络正确归类的语句样本. 表 2 中的结果表明, 在 CogCompQC 模型上本方法相较于 POPQORN 求得的鲁棒半径亦有明显提升, 均值提升范围在 27.3%–30.2% 不等. 注意到这几种模型的层数高于 MNIST 模型, 这一数据亦说明在网络层数增加时本方法求得的结果精度提升也随之提高.

表 2 CogCompQC 数据集鲁棒半径对比

网络模型	距离度量	POPQORN (最大值 | 最小值 | 平均值)	VRNN-LD (最大值 | 最小值 | 平均值)	均值提升率 (%)
RNN×32×16	l_1	0.3148 | 0.0686 | 0.1811	0.3978 | 0.0744 | 0.2325	28.3
	l_2	0.0946 | 0.0351 | 0.0427	0.1022 | 0.0367 | 0.0544	27.6
	l_∞	0.0134 | 0.0023 | 0.0079	0.0153 | 0.0088 | 0.0102	30.2
RNN×64×16	l_1	0.2907 | 0.0572 | 0.1790	0.3072 | 0.0617 | 0.2278	27.3
	l_2	0.0835 | 0.0237 | 0.0382	0.0908 | 0.0383 | 0.0493	29.1
	l_∞	0.0141 | 0.0009 | 0.0093	0.0229 | 0.0025 | 0.0120	29.6

5.2 LSTM 鲁棒半径计算实验与分析

关于 LSTM 模型鲁棒半径计算方法的性能评估与分析, 实验配置与上文类似, 在 MNIST 数据集训练的 LSTM 模型上进行. 考虑到 LSTM 模型结构较朴素 RNN 复杂, 实验采用的 6 种模型其层数为 2–7 不等, 隐藏层尺寸仍为 32 或 64. 实验中考虑到采样点对求解精度的影响, 将 VRNN-LD 计算限界平面参数时的采样点数 N 设置为 70, 以在采样精度和计算成本之间取得平衡, 对偶变量的更新步长设置与朴素 RNN 相同. 表 3 给出两种方法在不同 LSTM 模型及多种距离度量下求得的鲁棒半径的详细结果.

表 3 MNIST 数据集鲁棒半径对比 (LSTM)

网络模型	距离度量	POPQORN (最大值 | 最小值 | 平均值)	VRNN-LD (最大值 | 最小值 | 平均值)	均值提升率 (%)
LSTM×32×2	l_1	1.9314 | 0.3885 | 1.0550	2.0508 | 0.4107 | 1.1102	5.2
	l_2	0.2872 | 0.1486 | 0.2033	0.2887 | 0.1492 | 0.2206	8.5
	l_∞	0.0347 | 0.0059 | 0.0195	0.0377 | 0.0064 | 0.0213	9.2
LSTM×64×2	l_1	2.4652 | 0.6783 | 1.3197	2.6253 | 0.6824 | 1.4003	6.1
	l_2	0.4328 | 0.1480 | 0.2461	0.5181 | 0.1509 | 0.2651	7.7
	l_∞	0.0311 | 0.0134 | 0.0219	0.0458 | 0.0141 | 0.0235	7.3
LSTM×32×4	l_1	1.4656 | 0.3189 | 0.8342	1.7531 | 0.3263 | 0.8832	5.9
	l_2	0.2818 | 0.0913 | 0.1751	0.2841 | 0.0943 | 0.1848	5.5
	l_∞	0.0284 | 0.0079 | 0.0162	0.0340 | 0.0094 | 0.0173	6.2
LSTM×64×4	l_1	1.6781 | 0.2459 | 0.9544	1.9863 | 0.2614 | 1.0212	7.0
	l_2	0.3654 | 0.0981 | 0.2167	0.3830 | 0.1137 | 0.2315	6.8
	l_∞	0.0309 | 0.0085 | 0.0184	0.0314 | 0.0091 | 0.0196	6.5
LSTM×32×7	l_1	0.5480 | 0.1520 | 0.3046	0.5782 | 0.1795 | 0.3261	7.1
	l_2	0.1011 | 0.0584 | 0.0792	0.1039 | 0.0616 | 0.0873	10.2
	l_∞	0.0173 | 0.0047 | 0.0101	0.0219 | 0.0049 | 0.0109	7.9
LSTM×64×7	l_1	0.7084 | 0.2671 | 0.4018	0.7488 | 0.2875 | 0.4332	7.8
	l_2	0.1346 | 0.0762 | 0.0993	0.1474 | 0.0766 | 0.1086	9.4
	l_∞	0.0201 | 0.0081 | 0.0126	0.0208 | 0.0103 | 0.0137	8.7

从表 3 所示的实验数据可以看出, VRNN-LD 求解出的鲁棒半径在 6 类 LSTM 模型以及 3 种范数扰动类型下, 平均值相较于 POPQORN 提升了 5.2%–10.2% 不等. 由于 LSTM 中存在非线性激活函数交叉相乘的复杂情形,

对其进行限界平面近似难以根据待近似曲面的特点进行进一步的细化, 因此相较于朴素 RNN 中不同情况下的 tanh 近似而言, 结果的提升较为有限.

表 4 给出了 CogCompQC 数据集鲁棒半径求解结果的对比. 如表 4 所示, VRNN-LD 求解出的鲁棒半径在层数为 12 的两种问题分类网络上, 平均值相较于 POPQORN 提升范围为 5.4%–9.3% 不等. 这一结果与上述 MNIST 模型的结果相比提升率相近, 即随着网络层数的增加鲁棒半径的提升率保持稳定. 综上所述, 本方法在验证 LSTM 网络时有着一定的精度优势.

表 4　CogCompQC 数据集鲁棒半径对比 (LSTM)

网络模型	距离度量	POPQORN (最大值 | 最小值 | 平均值)	VRNN-LD (最大值 | 最小值 | 平均值)	均值提升率 (%)
LSTM×32×12	l_1	0.4863 | 0.2192 | 0.3594	0.4982 | 0.2231 | 0.3835	6.7
	l_2	0.2042 | 0.0781 | 0.1125	0.2177 | 0.0830 | 0.1186	5.4
	l_∞	0.0257 | 0.0111 | 0.0173	0.0284 | 0.0115 | 0.0185	7.1
LSTM×64×12	l_1	0.6873 | 0.2684 | 0.4479	0.7389 | 0.2841 | 0.4896	9.3
	l_2	0.2323 | 0.1137 | 0.1612	0.2522 | 0.1146 | 0.1712	6.2
	l_∞	0.0298 | 0.0155 | 0.0214	0.0342 | 0.0168 | 0.0232	8.4

5.3　验证时间与可扩展性评估

对于鲁棒半径求解方式的神经网络验证方法而言, 求得的鲁棒半径大小是首要的评估指标, 反映了方法求解验证问题的能力. 然而, 也有必要对验证时间进行实验分析, 以评估方法的可扩展性. 表 5 给出了两种方法在部分朴素 RNN 模型和 LSTM 模型上的样本平均验证时间对比, 时间单位为秒. 模型选用了规模相对较大的 6 种 RNN 模型和 6 种 LSTM 模型, 涵盖了 MNIST 和 CogCompQC 两个数据集. 模型规模指神经元数量乘以层数.

表 5　验证时间对比

网络模型	距离度量	POPQORN (s)	VRNN-LD (s)	耗时增加 (%)	网络模型	距离度量	POPQORN (s)	VRNN-LD (s)	耗时减少 (%)
RNN×32×7	l_1	0.239	0.716	199.6	LSTM×32×4	l_1	178.823	73.032	59.2
	l_2	0.237	0.753	217.7		l_2	237.648	90.682	61.8
	l_∞	0.261	0.795	204.6		l_∞	255.745	107.391	58.0
RNN×64×7	l_1	0.321	0.877	173.2	LSTM×64×4	l_1	230.206	90.510	60.7
	l_2	0.283	0.817	188.7		l_2	269.811	105.609	60.9
	l_∞	0.268	0.842	214.2		l_∞	279.171	119.298	57.3
RNN×32×14	l_1	0.461	1.281	177.9	LSTM×32×7	l_1	270.631	103.909	61.6
	l_2	0.416	1.259	202.6		l_2	377.289	154.884	58.9
	l_∞	0.503	1.388	175.9		l_∞	424.963	178.197	58.1
RNN×64×14	l_1	0.561	1.695	202.1	LSTM×64×7	l_1	363.354	160.897	55.7
	l_2	0.539	1.713	217.8		l_2	466.015	207.103	55.6
	l_∞	0.710	1.944	173.8		l_∞	495.117	281.274	43.2
RNN×32×16	l_1	0.481	1.416	194.4	LSTM×32×12	l_1	802.418	294.971	63.2
	l_2	0.532	1.634	207.1		l_2	862.338	301.071	65.1
	l_∞	0.674	1.940	192.9		l_∞	1 002.173	401.087	60.0
RNN×64×16	l_1	0.621	1.899	205.8	LSTM×64×12	l_2	1 279.247	468.833	63.4
	l_2	0.626	1.982	216.6		l_2	1 278.683	495.062	61.3
	l_∞	0.817	2.378	191.1		l_∞	1 159.264	596.545	48.5

如表 5 中的实验结果所示, 在朴素 RNN 模型上, 本文方法 VRNN-LD 的验证时间高于现有方法 POPQORN, 耗时增加 173.2%–217.8% 不等. 耗时相对较多的原因是方法对 S 型激活函数的近似采用了两条上界或下界的方

式, 需要避免边界数量在逐层传播中组合爆炸, 因此相较于 POPQORN 的单上界/下界近似技术精度更高但求解更难. 此外, 注意到耗时增加的幅度比较稳定, 并未随着模型规模的增加而增加, 且上述验证时间均在数秒之内, 这一耗时增加是可接受的. 由此换来的精度提升对于鲁棒半径求解方法而言更为重要.

另一方面, 在 LSTM 模型上, VRNN-LD 的耗时低于 POPQORN 方法, 耗时减少 43.2%–65.1% 不等. 原因是采用了效率较高的采样法对二元非线性曲面进行上下界近似, 可简化近似平面相关的计算. 此外, 注意到 LSTM 模型的验证时间远大于规模相近的朴素 RNN 模型, 且随着层数的增加验证时间的增长明显. 体现出 LSTM 的结构相较于朴素 RNN 处理难度更大, 对 LSTM 模型的验证效率亦成为 RNN 验证方法的瓶颈.

最后对方法的可扩展性进行评估. 对于神经网络验证方法而言精确评估非常难, 故此处仅做大致评估. 如表 5 所示, VRNN-LD (同 POPQORN) 能够有效处理规模在 1 000 以内的朴素 RNN 模型, 且仍有一定的扩展空间. 注意到 RNN×64×16 的规模为 1 000 左右, 能够在数秒内被验证. 对于 LSTM 模型, VRNN-LD (同 POPQORN) 能够处理的模型规模为数百以内. 注意到 LSTM×64×12 模型的验证耗时较长, 几乎达到可接受的极限. 总体而言, 本文方法的可扩展性与现有方法 POPQORN 相当, 主要适用于规模较小的 RNN 模型的验证.

6 总 结

本文提出了一种基于线性松弛与对偶优化的循环神经网络验证方法 VRNN-LD. 该方法整体属于近似求解, 对 RNN 中的非线性激活函数根据不同情况采用更加精确的线性松弛, 将网络输出层边界求解的优化问题进行拉格朗日分解, 根据分解后子问题的解来逐步优化对偶变量以得到输出层的上下界, 并在此基础上采用二分法对网络最终的鲁棒半径进行求解. 实验结果表明, VRNN-LD 较已有同类型方法在不同 RNN 模型和多种范数度量扰动下均能求解得到更大的鲁棒半径.

后续工作计划对 GRU 及更多扩展类型的 RNN 模型验证进行探索. 如何针对结构更为复杂、被广泛应用于现实场景下的网络设计出可行且精确的鲁棒半径求解方法是今后研究的挑战.

References

[1] Pouyanfar S, Sadiq S, Yan YL, Tian HM, Tao YD, Reyes MP, Shyu ML, Chen SC, Iyengar SS. A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys, 2019, 51(5): 92. [doi: 10.1145/3234150]

[2] Rao Q, Frtunikj J. Deep learning for self-driving cars: Chances and challenges. In: Proc. of the 1st Int'l Workshop on Software Engineering for AI in Autonomous Systems. Gothenburg: ACM, 2018. 35–38. [doi: 10.1145/3194085.3194087]

[3] Al-Salman O, Mustafina J, Shahoodh G. A systematic review of artificial neural networks in medical science and applications. In: Proc. of the 13th Int'l Conf. on Developments in eSystems Engineering (DeSE). Liverpool: IEEE, 2020. 279–282. [doi: 10.1109/DeSE51703.2020.9450245]

[4] Ji SL, Du TY, Deng SG, Cheng P, Shi J, Yang M, Li B. Robustness certification research on deep learning models: A survey. Chinese Journal of Computers, 2022, 45(1): 190–206 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2022.00190]

[5] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proc. of the 2017 IEEE Symp. on Security and Privacy (SP). San Jose: IEEE, 2017. 39–57. [doi: 10.1109/SP.2017.49]

[6] Biggio B, Corona I, Maiorca D, Nelson B, Šrndić N, Laskov P, Giacinto G, Roli F. Evasion attacks against machine learning at test time. In: Proc. of the 2013 European Conf. on Machine Learning and Knowledge Discovery in Databases. Prague: Springer, 2013. 387–402. [doi: 10.1007/978-3-642-40994-3_25]

[7] Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao CW, Prakash A, Kohno T, Song D. Robust physical-world attacks on deep learning visual classification. In: Proc. of the 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Lake City: IEEE, 2018. 1625–1634. [doi: 10.1109/CVPR.2018.00175]

[8] Liu CL, Arnon T, Lazarus C, Strong C, Barrett C, Kochenderfer MJ. Algorithms for verifying deep neural networks. Foundations and Trends in Optimization, 2021, 4(3–4): 244–404. [doi: 10.1561/2400000035]

[9] Liu ZX, Yang PF, Zhang LJ, Wu ZL, Huang XW. Survey on acceleration techniques for complete neural network verification. Ruan Jian Xue Bao/Journal of Software, 2024, 35(9): 4038–4068 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7127.htm [doi: 10.13328/j.cnki.jos.007127]

[10] Dong YS, Liu YH, Dong XQ, Zhao L, Tian C, Yu B, Duan ZH. Verification for neural network based on error divide and conquer. Ruan

Jian Xue Bao/Journal of Software, 2024, 35(5): 2307–2324 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6967.htm [doi: 10.13328/j.cnki.jos.006967]

[11] Wang SQ, Pei KX, Whitehouse J, Yang JF, Jana S. Efficient formal safety analysis of neural networks. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 6369–6379.

[12] Guo XW, Zhou ZW, Zhang YL, Katz G, Zhang M. OccRob: Efficient SMT-based occlusion robustness verification of deep neural networks. In: Proc. of the 29th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Paris: Springer, 2023. 208–226. [doi: 10.1007/978-3-031-30823-9_11]

[13] Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ. Reluplex: An efficient SMT solver for verifying deep neural networks. In: Proc. of the 29th Int'l Conf. on Computer Aided Verification (CAV). Heidelberg: Springer, 2017. 97–117. [doi: 10.1007/978-3-319-63387-9_5]

[14] Fischetti M, Jo J. Deep neural networks and mixed integer linear optimization. Constraints, 2018, 23(3): 296–309. [doi: 10.1007/s10601-018-9285-6]

[15] Kouvaros P, Lomusci, A. Towards scalable complete verification of ReLU neural networks via dependency-based branching. In: Proc. of the 30th Int'l Joint Conf. on Artificial Intelligence (IJCAI). Montreal: ijcai.org, 2021. 2643–2650. [doi: 10.24963/ijcai.2021/364]

[16] Singh G, Gehr T, Püschel M, Vechev M. An abstract domain for certifying neural networks. Proc. of the ACM on Programming Languages, 2019, 3(POPL): 41. [doi: 10.1145/3290354]

[17] Weng TW, Zhang H, Chen HG, Song Z, Hsieh CJ, Daniel L, Boning DS, Dhillon IS. Towards fast computation of certified robustness for ReLU networks. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 5273–5282.

[18] Wu HZ, Barrett C, Sharif M, Narodytska N, Singh G. Scalable verification of GNN-based job schedulers. Proc. of the ACM on Programming Languages, 2022, 6(OOPSLA2): 1036–1065. [doi: 10.1145/3563325]

[19] Dathathri S, Dvijotham K, Kurakin A, Raghunathan A, Uesato J, Bunel R, Shankar S, Steinhardt J, Goodfellow I, Liang P, Kohli P. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 447.

[20] Fazlyab M, Morari M, Pappas GJ. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Trans. on Automatic Control, 2022, 67(1): 1–15. [doi: 10.1109/TAC.2020.3046193]

[21] Boopathy A, Weng TW, Chen PY, Liu S, Daniel L. CNN-Cert: An efficient framework for certifying robustness of convolutional neural networks. In Proc. of the 33rd AAAI Conf. on Artificial Intelligence. Honolulu: AAAI Press, 2019. 3240–3247. [doi: 10.1609/AAAI.V33I01.33013240]

[22] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735–1780. [doi: 10.1162/NECO.1997.9.8.1735]

[23] Elman JL. Finding structure in time. Cognitive Science, 1990, 14(2): 179–211. [doi: 10.1207/s15516709cog1402_1]

[24] Ko CY, Lyu ZY, Weng L, Daniel L, Wong N, Lin DH. POPQORN: Quantifying robustness of recurrent neural networks. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 3468–3477.

[25] Ehlers R. Formal verification of piece-wise linear feed-forward neural networks. In: Proc. of the 15th Int'l Symp. Automated Technology for Verification and Analysis (ATVA). Pune: Springer, 2017. 269–286. [doi: 10.1007/978-3-319-68167-2_19]

[26] Narodytska N, Kasiviswanathan S, Ryzhyk L, Sagiv M, Walsh T. Verifying properties of binarized deep neural networks. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI Press, 2018. 6615–6624. [doi: 10.1609/AAAI.V32I1.12206]

[27] Cheng CH, Nührenberg G, Ruess H. Maximum resilience of artificial neural networks. In: Proc. of the 15th Int'l Symp. Automated Technology for Verification and Analysis (ATVA). Pune: Springer, 2017. 251–268. [doi: 10.1007/978-3-319-68167-2_18]

[28] Wang SQ, Pei KX, Whitehouse J, Yang JF, Jana S. Formal security analysis of neural networks using symbolic intervals. In: Proc. of the 27th USENIX Conf. on Security Symp. Baltimore: USENIX Association, 2018. 1599–1614.

[29] Zhang H, Weng TW, Chen PY, Hsieh CJ, Daniel L. Efficient neural network robustness certification with general activation functions. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 4944–4953.

[30] Wang SQ, Zhang H, Xu KD, Lin X, Jana S, Hsieh CJ, Kolter JZ. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 2289.

[31] Zhang H, Wang SQ, Xu KD, Li LY, Li B, Jana S, Hsieh CJ, Kolter JZ. General cutting planes for bound-propagation based neural network verification. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 121.

[32] Singh G, Gehr T, Mirman M, Püschel M, Vechev M. Fast and effective robustness certification. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 10825–10836.

[33] Singh G, Ganvir R, Püschel M, Vechev M. Beyond the single neuron convex barrier for neural network certification. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 1352.

[34] Wong E, Kolter Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholmsmässan: PMLR, 2018. 5283–5292.

[35] Gowal S, Dvijotham K, Stanforth R, Mann T, Kohli P. A dual approach to verify and train deep networks. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence (IJCAI). Macao: ijcai.org, 2019. 6156–6160. [doi: 10.24963/ijcai.2019/854]

[36] Bunel R, de Palma A, Desmaison A, Dvijotham K, Kohli P, Torr PHS, Kumar MP. Lagrangian decomposition for neural network verification. In: Proc. of the 36th Conf. on Uncertainty in Artificial Intelligence (UAI). AUAI Press, 2020. 370–379.

[37] Du TY, Ji SL, Shen LJ, Zhang Y, Li JF, Shi J, Fang CF, Yin JW, Beyah R, Wang T. Cert-RNN: Towards certifying the robustness of recurrent neural networks. In: Proc. of the 2021 ACM SIGSAC Conf. on Computer and Communications Security (CCS). ACM, 2021. 516–534. [doi: 10.1145/3460120.3484538]

[38] Wu YT, Zhang M. Tightening robustness verification of convolutional neural networks with fine-grained linear approximation. In: Proc. of the 35th AAAI Conf. on Artificial Intelligence. AAAI Press, 2021. 11674–11681. [doi: 10.1609/AAAI.V35I13.17388]

[39] Guignard M, Kim S. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. Mathematical Programming, 1987, 39(2): 215–228. [doi: 10.1007/BF02592954]

[40] Li X, Roth D. Learning question classifiers: The role of semantic information. Natural Language Engineering, 2006, 12(3): 229–249. [doi: 10.1017/S1351324905003955]

附中文参考文献

[4] 纪守领, 杜天宇, 邓水光, 程鹏, 时杰, 杨珉, 李博. 深度学习模型鲁棒性研究综述. 计算机学报, 2022, 45(1): 190–206. [doi: 10.11897/SP.J.1016.2022.00190]

[9] 刘宗鑫, 杨鹏飞, 张立军, 吴志林, 黄小炜. 完备神经网络验证加速技术综述. 软件学报, 2024, 35(9): 4038–4068. http://www.jos.org.cn/1000-9825/7127.htm [doi: 10.13328/j.cnki.jos.007127]

[10] 董彦松, 刘月浩, 董旭乾, 赵亮, 田聪, 于斌, 段振华. 基于误差分治的神经网络验证. 软件学报, 2024, 35(5): 2307–2324. http://www.jos.org.cn/1000-9825/6967.htm [doi: 10.13328/j.cnki.jos.006967]

作者简介

赵亮, 博士, 副教授, CCF 专业会员, 主要研究领域为形式化方法, 神经网络的形式化验证, 时序逻辑.

杨成龙, 硕士生, 主要研究领域为神经网络的形式化验证.

闫宗祥, 硕士生, 主要研究领域为神经网络的形式化验证.

王小兵, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为形式化方法, 时序逻辑, 性质挖掘.