基于健壮半径求解的循环神经网络形式化验证方法

赵 亮[1], 戚润川[1], 段鑫民[1], 李春奕[1], 王小兵[1*]

[1]西安电子科技大学计算机科学与技术学院 西安 中国 710071

摘要 随着软硬件技术的发展, 神经网络在各行各业取得了广泛的应用, 然而在应用过程中也暴露出健壮性的不足。因此, 采用形式化的手段来检验和保障神经网络的安全性是至关重要的。然而, 由于循环神经网络结构复杂、激活函数非线性等因素, 目前关于这类神经网络的形式化验证工作非常有限。针对循环神经网络难于验证的问题, 本文提出了 VR-RRS, 一种基于健壮半径求解的循环神经网络形式化验证方法。首先, 基于神经网络验证的经典近似求解框架, 通过逐层回溯迭代的方式得到循环神经网络各层神经元近似区间上下界关于输入的线性表达式, 在此基础上利用赫尔德不等式推导出各层神经元的近似上下界关于扰动半径的解析解。随后, 针对经典近似求解方法精度不高的问题, 通过对激活函数的近似方式进行分析和优化, 提出一种基于多路径回溯的求解策略。该策略以细粒度近似方法构造不同的回溯路径, 在此基础上将这些回溯路径求解的近似区间取交集, 能够得到更为精确的近似区间。最后, 采用改进的二分法对健壮半径进行求解, 主要针对经典二分法中精度不足的问题, 优化了判断神经网络健壮性的标准。通过在不同结构的循环神经网络上与已有方法开展对比实验, 结果表明了该方法在求解出的健壮半径和验证成功率上具有明显优势。

关键词 形式化验证; 循环神经网络; 健壮半径; 近似求解

中图法分类号 TP311 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2023.05.02

A Formal Verification Method for Recurrent Neural Network Based on Robustness Radius Solving

ZHAO Liang[1], QI Runchuan[1], DUAN Xinmin[1], LI Chunyi[1], WANG Xiaobing[1]

[1]School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Abstract With the development of software and hardware technology, neural networks have been widely applied to the various walks of life, but they are also exposed with insufficient robustness during their applications. Therefore, it is vital to use formal methods to check and ensure the security of neural networks. However, due to the facts such as complex structure and nonlinear activation function of recurrent neural networks, the work on formal verification of this kind of neural networks is very limited at present. Aiming at the problem that recurrent neural networks are difficult to verify, this paper proposes VR-RRS, a formal Verification method for Recurrent neural networks based on Robustness Radius Solution. First, based on the classical approximate solution framework of neural network verification, the linear expressions of the upper and lower bounds of the approximate interval of neurons at each layer about the input of the recurrent neural network are obtained through layer-by-layer backtracking iteration. On this basis, the analytic solutions of the upper and lower bounds of the approximate interval of neurons at each layer about the disturbance radius are derived using Holder inequality. Then, aiming at the problem that the accuracy of the classical approximate solving method is not high, by analyzing and optimizing approximation modes of the activation function, a solving strategy based on multi-path backtracking is proposed. This strategy constructs different backtracking paths with the fine-grained approximation method. On this basis, the approximate intervals solved by these backtracking paths are intersected to obtain a more accurate approximate interval. Finally, an improved dichotomy is used to solve the robust radius, which mainly aims at the problem of insufficient precision in the classical dichotomy and optimizes the standard to judge the robustness of neural networks. Through comparative experiments with existing methods on recurrent neural networks with different structures, this method shows obvious advantages in the robustness radius obtained and the success rate of verification.

Key words formal verification; recurrent neural network; robustness radius; approximate solving

通讯作者: 王小兵, 博士, 副教授, Email: xbwang@mail.xidian.edu.cn。

本课题得到国家自然科学基金(No. 61972301, No. 61672403, No. 62192730, No. 62192734); 西安市科技计划项目(No. 22GXFW0025); 陕西省重点研发计划(No. 2020GY-043); 陕西省重点科技创新团队(No. 2019TD-001)资助。

收稿日期: 2022-09-10; 修改日期: 2022-12-14; 定稿日期: 2023-03-28

1 引言

近年来, 人工智能引领科学技术的发展变革, 也给人们的生活带来了天翻地覆的变化。其中受到广泛关注的神经网络领域涌现出很多表现出色的模型, 比如卷积神经网络[1]、循环神经网络[2-3]等。借助于这些神经网络模型以及各种优化学习技术, 可以对现实物理世界中高度抽象的关系进行近似。因此, 它们在在图像识别、自然语言处理等一些输入输出关系过于复杂而难以精确建模的任务中取得了广泛的应用[4-6]。

虽然神经网络在诸多领域展示了其优异的性能, 但其始终无法绝对精确地对无穷输入空间到输出空间的映射关系进行建模, 且模型自身仍然存在着健壮性差、行为不透明、可解释性差[7]等不可忽视的问题, 因此在实际应用中因神经网络系统的误判造成的事故屡见不鲜。例如, 2016 年 5 月, 美国的一名司机在驾驶特斯拉 Model-S 的过程中, 开启了自动驾驶功能, 因识别系统的误判撞上了道路中央的半挂卡车, 当场死亡。同时学术界先后提出了基于梯度、基于迭代和基于生成对抗网络的攻击方法[8-10]。这些方法成功地对各种优秀的神经网络模型进行攻击, 证明了即使在输入上施加极为微小的扰动, 也可能会导致神经网络误判, 在理论研究层面暴露了神经网络的脆弱性。

基于上述背景, 人们开展了对神经网络安全可信性的相关研究, 旨在判定或保障这一类系统的安全性, 并为此探索了多种技术。其中, 相比于传统的测试技术, 形式化验证技术可以为系统安全提供可靠与正确的保证。因此, 神经网络的形式化验证取得了广泛关注。目前的神经网络验证技术大致可分为两类: 精确求解方法[11-16]和近似求解方法[17-21]。精确求解方法主要是基于离散优化理论来形式化验证神经网络中某些属性对于任何可能的输入的可信性, 即利用可满足性模理论(Satisfiability Modulo Theories, SMT)或混合整数线性规划(Mixed Integer Linear Programming, MILP)来解决此类形式验证问题。这类方法通常是通过利用 ReLU 的分段线性特性并在搜索可行解时尝试逐渐满足它们施加的约束来实现的。基于可满足性模理论的研究工作主要以 Reluplex[11]为代表, 该工作通过扩展单纯形算法以支持 ReLU 约束, 并验证了激活函数为 ReLU 的前馈神经网络的健壮性。Lomuscio 等人[13]提出的方法将神经网络编码成混合整数线性规划问题, 对线性规划中的浮点运算处理方式进行优化, 并采用了一种较为高效的线性规划求解方法, 相比于 Reluplex 中作为对比的线性规划方法效率大大提升。精确求解方法具有完备性, 能够返回一个确定性结果, 但这类方法具有很高的计算复杂度, 只适用于使用 ReLU 这样的分段线性激活函数的浅层小规模网络。

一些学者认为, 精确求解的方法将神经网络验证问题考虑得过于全面, 导致问题复杂度过高, 时间开销过大, 可以适当地将验证问题简化, 采用近似的方法进行求解[17-21]。这类方法对神经网络进行健壮性验证通常有一个一般性框架, 即在给定扰动类型与扰动范围的情况下, 通过将激活函数抽象为包含其凸包的线性约束区域, 可以逐层回溯迭代计算神经网络的近似区间, 并通过对输出层的近似区间进行分析来判断神经网络是否具有健壮性。同时为了量化健壮性, 在计算出输出层可达区间的基础上, 可以通过二分法得到一个扰动范围的健壮半径, 确保在此半径范围内所有样本分类正确。这一类型的代表工作有 Fast-Lin[19]、CROWN[20]和 DeepPoly[21]。Fast-Lin[19]给出了一套通过线性近似来计算神经元近似区间的流程, 针对非线性激活函数 ReLU 采用平行近似方法。CROWN[20]在 Fast-Lin 的基础上对 ReLU 提出了效果更好的自适应近似, 同时增加了对 ReLU 以外的激活函数的支持。DeepPoly[21]是一种基于抽象解释的方法, 这种方法面向神经网络验证设计了一种专门的子多面体抽象域, 主要思想是为神经元维护一个具体值的区间、一个符号化约束上界与一个符号化约束下界, 并且针对仿射变换与最大池化操作分别设计了对应的抽象转换操作。近似求解不能保证在任何情况下都给出确定的验证结论, 往往只能给出单边性的保障。然而, 相较于精确求解方法, 近似求解方法具有较低的时间复杂度, 适用于验证规模相对较大的网络。

现有的神经网络形式化验证的研究, 大多面向使用 ReLU 激活函数的全连接或卷积神经网络[22], 关于循环神经网络验证的工作非常少[23-24]。而循环神经网络已被广泛应用于语音识别、自然语言处理、序列生成等多种现实场景, 其安全性的检验和保障具有重要的意义[25-27]。在此背景下, 本文旨在对循环神经网络的健壮性进行验证, 以改进目前该领域研究工作不足这一现状。然而, 循环神经网络的健壮性验证是一个复杂的问题。这类网络通常用来处理序列的建模, 网络层次较深, 且使用 tanh 等非分段线性激活函数。针对这一问题, 采用近似求解的方法更为适合。

本文提出了 VR-RRS (Verification of RNN based

on Robustness Radius Solving), 一种基于健壮半径求解的循环神经网络形式化验证方法。首先, 该方法在经典近似求解框架上进行了改进, 引入细粒度的激活函数近似方式, 并将其应用于循环神经网络近似上下界的求解。其次, 分析细粒度激活函数近似方式的优势以及存在的问题, 并在此基础上设计了基于多路径回溯的近似区间求解方法, 使得循环神经网络各层的近似区间进一步精确。同时, 改进了传统近似求解框架的二分法求解过程。不同于传统方法中将输出层标签神经元的近似区间下界与其他神经元近似区间上界进行直接比较, 该方法直接计算标签神经元与其他神经元之差的近似区间。这种方式减小了因不等式放缩带来的对健壮性判定结果影响, 从而提高求得的健壮半径, 即失真认证下界。本文在基于 MNIST 数据集的不同结构的循环神经网络上与同类方法 POPQORN[23]进行了对比实验。实验结果表明, VR-RRS 可以大幅度提升求得的健壮半径, 并且在验证成功率上具有明显优势。

本文的内容安排如下。第 2 节介绍相关背景和概念。第 3 节介绍本文提出的基于健壮半径求解的循环神经网络验证方法 VR-RRS。第 4 节通过对比实验对所提出的方法进行评估。第 5 节总结全文。

2 背景知识

2.1 循环神经网络

循环神经网络是一种具有递归结构的神经网络。这种特殊的结构可以记忆上一个输入序列的特性, 因此这种网络善于从具有一定顺序意义的样本与样本间学习规律, 适用于解决连续序列相关的问题, 如自然语言处理。循环神经网络的主要计算过程用数学公式展示如下:

$$a^{(k)}=\tanh\left(W^{\mathrm{aa}}a^{(k)}+W^{\mathrm{ax}}x^{(k)}+b^{\mathrm{a}}\right) \tag{1}$$

$$F(X)=W^{\mathrm{Fa}}a^{(m)}+b^{\mathrm{F}} \tag{2}$$

式(1)表示循环神经网络隐藏层的状态传递, 其中 $a^{(k)}$ 代表第 k 层的隐藏层状态, W^{aa} 代表隐藏层与隐藏层之间的权值矩阵, $x^{(k)}$ 代表第 k 层的输入, W^{ax} 代表连接隐藏层与输入层的权重, b^{a} 代表隐藏层的偏置。从式(1)中可以看出, 循环神经网络的隐藏层状态是由当前层的输入和上一层的隐藏层状态所决定的, 这体现了循环神经网络的递归结构与记忆性质。

式(2)是循环神经网络的输出层计算公式。因为一般神经网络健壮性验证问题针对的是分类网络, 所以本文选取多对一结构的用于分类的循环神经网络, 即只有最后一层拥有输出的循环神经网络。式(2)中, m 代表循环神经网络的层数, F 是一个代表循环神经网络的函数, W^{Fa} 代表输出层权重, b^{F} 代表输出层偏置, X 是整个网络的总体输入。一个简单的循环神经网络的结构示例如图 1 所示。

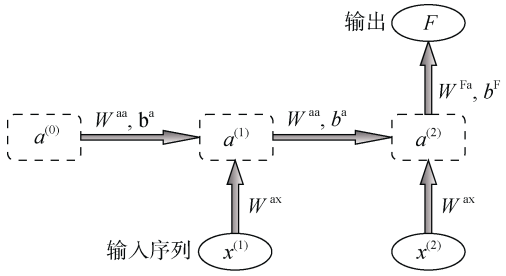

图 1 一个层数为 2 的多对一结构循环神经网络

Figure 1 A many-to-one recurrent neural network with 2 layers

2.2 健壮性验证问题定义

在给出循环神经网络的健壮性验证相关问题定义之前, 首先给出一些符号约定。令 $X_0=\left[x_0^{(1)},\ldots,x_0^{(m)}\right]$ 为循环神经网络的一个原始输入序列, $\widetilde{X}=\left[\tilde{x}^{(1)},\ldots,\tilde{x}^{(m)}\right]$ 为原始序列一个确定的扰动序列, label(X) 表示输入序列的标签, $f(X)$ 代表输入序列经循环神经网络处理后得到的分类结果。

定义 1(健壮性验证问题)给定一个距离度量 p 和扰动距离值 ϵ, 健壮性验证问题的目标是确定是否对于任意一个输入序列 $\widetilde{X}$, 满足 $\tilde{x}^{(k)}\in\mathbb{B}_p\left(x_0^{(k)},\epsilon\right)$, 有 $\mathrm{label}(X_0)=f(\widetilde{X})$。

上述定义中, $\tilde{x}^{(k)}\in\mathbb{B}_p\left(x_0^{(k)},\epsilon\right)$ 表示 $\tilde{x}^{(k)}$ 满足 $\left\|\tilde{x}^{(k)}-x_0^{(k)}\right\|_p\leqslant\epsilon$, 其中 $\mathbb{B}_p\left(x_0^{(k)},\epsilon\right)$ 表示在距离度量 p 下一个半径为 ϵ 的球状空间。由定义 1 可以看出, 健壮性验证问题是为了判断在距离原始输入序列给定范围的空间内是否存在分类结果不同的样本, 即对抗样本。如果空间内所有样本的分类结果与原始样本 X_0 相同, 网络具有健壮性。否则, 如果空间内存在对抗样本, 那么网络不健壮。如何量化健壮性也是一个重要的问题, 接下来将介绍用来量化健壮性的一个重要指标: 健壮半径。健壮半径是指一个距离值, 与原始样本不超过此距离值的空间范围内所有样本的分类结果相同, 即不存在对抗样本。

定义 2(健壮半径)给定一个距离度量 p, 若扰动距离值 ϵ 满足 $\forall\tilde{x}^{(k)}\in\mathbb{B}_p\left(x_0^{(k)},\epsilon\right)$ $\mathrm{label}(X_0)=f(\widetilde{X})$,

则称 ϵ 是网络的一个健壮半径。

理论上存在健壮半径的上确界。然而, 文献[11]证明了即使是计算一个简单的仅使用 ReLU 激活函数神经网络的健壮半径上确界也是 NP 完全问题。基于近似求解方法实际求解出的健壮半径往往小于其上确界, 求解得到的健壮半径大小也是衡量近似求解方法优劣的重要标准。通常求解出的神经网络近似区间越精确, 能够求得的健壮半径越大。

2.3 激活函数线性近似

近似求解方式需要计算出近似区间关于输入的线性表达式, 然而神经网络常常采用非线性的激活函数。为此, 需要对激活函数进行线性松弛。线性松弛过程中上边界线与下边界线可分别用如下公式表示:

$$h_{\mathrm{U},r}^{(k)}(v)=\alpha_{\mathrm{U},r}^{(k)}\left(v+\beta_{\mathrm{U},r}^{(k)}\right) \tag{3}$$

$$h_{\mathrm{L},r}^{(k)}(v)=\alpha_{\mathrm{L},r}^{(k)}\left(v+\beta_{\mathrm{L},r}^{(k)}\right) \tag{4}$$

其中 $l_r^{(k)}\leqslant v\leqslant u_r^{(k)}$。$l_r^{(k)}$ 与 $u_r^{(k)}$ 分别代表第 k 层第 r 个神经元激活前值的近似区间的下界与上界。式(3)与式(4)定义了两个一元线性函数, 分别代表了上界线与下界线, 即对于 $l_r^{(k)}\leqslant v\leqslant u_r^{(k)}$ 有 $h_{\mathrm{L},r}^{(k)}(v)\leqslant \tanh(v)\leqslant h_{\mathrm{U},r}^{(k)}(v)$。同时, 两个线性函数的斜率与截距根据 $l_r^{(k)}$、$u_r^{(k)}$ 的大小得到, 即 $\alpha_{\mathrm{U},r}^{(k)}$、$\beta_{\mathrm{U},r}^{(k)}$、$\alpha_{\mathrm{L},r}^{(k)}$、$\beta_{\mathrm{L},r}^{(k)}$ 的值与 $l_r^{(k)}$、$u_r^{(k)}$ 相关。循环神经网络主要使用 tanh 激活函数。经典近似求解框架将激活函数 tanh 的上界线与下界线的确定分为三种情况, 如图 2 所示。(a) $l_r^{(k)}\geqslant 0$, 上边界线 $h_{\mathrm{U},r}^{(k)}$ 为区间 $[l_r^{(k)},u_r^{(k)}]$ 任意一点 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$ 的切线, 下边界线 $h_{\mathrm{L},r}^{(k)}$ 为左右两个端点的连线, 其中左端点为 $\left(l_r^{(k)},\tanh\left(l_r^{(k)}\right)\right)$, 右端点为 $\left(u_r^{(k)},\tanh\left(u_r^{(k)}\right)\right)$。(b) $u_r^{(k)}\leqslant 0$, 上边界线 $h_{U,r}^{(k)}$ 为左右两个端点的连线, 下边界线 $h_{\mathrm{L},r}^{(k)}$ 为区间 $[l_r^{(k)},u_r^{(k)}]$ 上任意一点 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$ 的切线。(c) $l_r^{(k)}<0<u_r^{(k)}$, 上边界线经过左端点 $\left(l_r^{(k)},\tanh\left(l_r^{(k)}\right)\right)$ 与 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$, 并且是点 $\left(d,\tanh\left(d\right)\right)$ 的切线。下边界线经过右端点 $\left(u_r^{(k)},\tanh\left(u_r^{(k)}\right)\right)$ 与 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$, 并且是点 $\left(d,\tanh\left(d\right)\right)$ 的切线。

Wu 等人[28]针对经典近似求解框架中存在的线性边界线对激活函数近似不够紧密的问题, 提出了一种细粒度的 tanh 激活函数线性近似方法。这种方法同样将激活函数的上界线与下界线的确定分为三种情况, 如图 3 所示。(a) $\tanh'(l_r^{(k)})<K<\tanh'(u_r^{(k)})$, 其中 K 为通用的记号 $K=\dfrac{\tanh\left(u_r^{(k)}\right)-\tanh\left(l_r^{(k)}\right)}{u_r^k-l_r^k}$。此时, 上边界线 $h_{\mathrm{U},r}^{(k)}$ 为左右两个端点的连线, 下边界线 $h_{\mathrm{L},r}^{(k)}$ 为区间 $[l_r^{(k)},u_r^{(k)}]$ 上某一点 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$ 的切线, 同时这条切线的斜率与上边界线相同。(b) $\tanh'(u_r^{(k)})<K<\tanh'(l_r^{(k)})$, 下边界线 $h_{\mathrm{U},r}^{(k)}$ 为左右两个端点的连线, 上边界线 $h_{\mathrm{L},r}^{(k)}$ 为区间 $[l_r^{(k)},u_r^{(k)}]$ 上某一点 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$ 的切线, 同时这条切线的斜率与下边界线相同。(c) $\tanh'(l_r^{(k)})<K$ 且 $\tanh'(u_r^{(k)})<K$, 此时上边界线经过左端点 $\left(l_r^{(k)},\tanh\left(l_r^{(k)}\right)\right)$ 与 $\left(d,\tanh\left(d\right)\right)$ $(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$, 并且是点 $\left(d,\tanh\left(d\right)\right)$

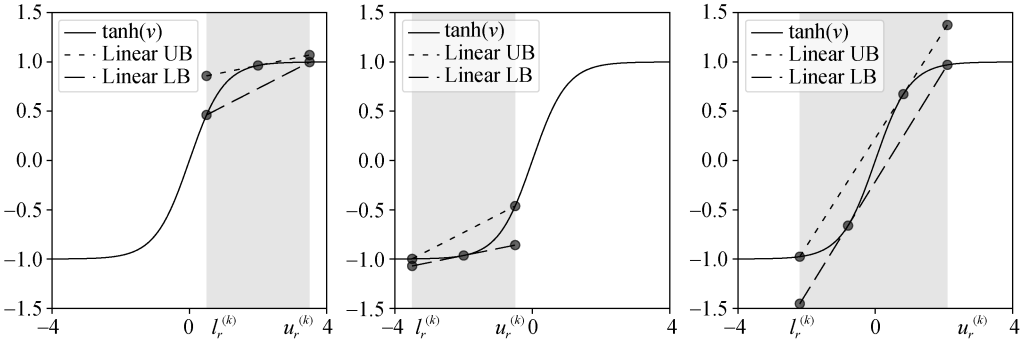

图 2 经典近似求解框架中的 tanh 激活函数线性近似方法

Figure 2 Linear approximation method of tanh activation function in the classical approximation framework

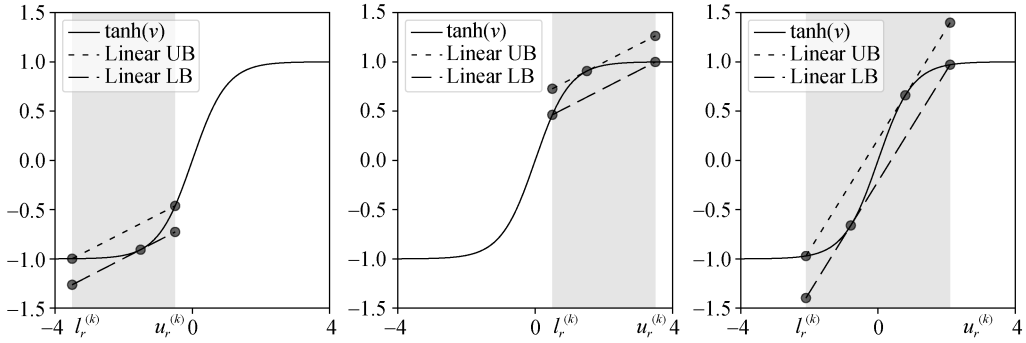

图 3　细粒度的 tanh 激活函数线性近似方法

Figure 3　Fine-grained linear approximation method of tanh activation function

的切线。下边界线经过右端点 $\left(u_r^{(k)},\tanh\left(u_r^{(k)}\right)\right)$ 与 $\left(d,\tanh\left(d\right)\right)(l_r^{(k)}\leqslant d\leqslant u_r^{(k)})$, 并且是点 $\left(d,\tanh\left(d\right)\right)$ 的切线。

与经典近似求解框架中的激活函数线性近似方法相比, 细粒度近似方法避免了个别情况下近似不够紧密的问题, 如图 4 所示。文献[27]通过在卷积神经网络进行健壮性验证的实验, 证明了基于细粒度近似方法可以求解出更紧密的近似区间, 同时可以获得较大的健壮半径。本工作受到该方法的启发, 并在此基础上进行了改进。

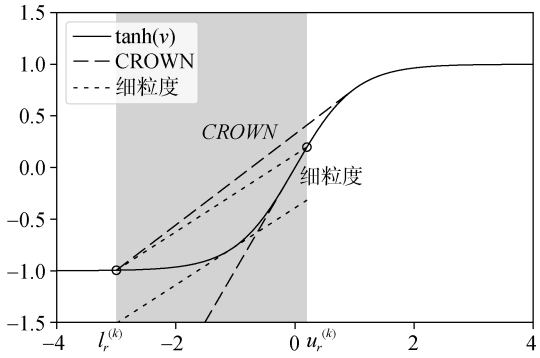

图 4　两种激活函数近似方法对比

Figure 4　Comparison of two approximation methods for the activation function

3　基于健壮半径求解的循环神经网络健壮性验证方法 VR-RRS

本节介绍所提出方法 VR-RRS 的具体流程。首先, 3.1 节对循环神经网络各层神经元的近似上下界进行推导。随后, 3.2 节基于多路径回溯策略求解更为精确的近似区间。在此基础上, 3.3 节采用一种改进的二分法对网络的健壮半径进行求解。

3.1　近似上下界公式推导

近似求解方式验证神经网络健壮性的一个关键点是在给出扰动范围后能够计算出神经元取值的上下界。本节给出循环神经网络各神经元近似上下界的具体推导过程与结果。为了方便阐述, 首先对一些相关概念进行说明。

参考经典近似求解框架, 通常只计算神经元激活前值(pre-activation)的近似上下界。对于循环神经网络而言, 需计算向量 $y^{(k)}=W^{\mathrm{aa}}a^{(k-1)}+W^{\mathrm{ax}}x^{(k)}+b^{\mathrm{a}}$ 中各元素值的近似上下界。注意到隐藏层与输出层的不同, 本文使用记号 $l_k^{(r)}$ 与 $u_k^{(r)}$ 分别代表第 k 层第 r 个神经元激活前值的下界与上界, 而使用记号 γ_j^{U} 与 γ_j^{L} 分别代表输出层第 j 个神经元取值的下界与上界。对这些近似上下界进行求解是一个逐层回溯迭代的过程。从数学推导层面来看, 需要各神经元的近似上下界回溯迭代至输入层, 从而得到关于输入的线性表达式, 再根据线性表达式得到关于扰动范围 ϵ 的解析表达式。为了能够得到神经元激活前值的上下界关于输入的线性表达式, 需要对非线性的激活函数进行线性松弛, 线性松弛过程如 2.3 节所示。

给定一个输出类别数为 t, 层数为 m, 隐藏层神经元数目为 s 的循环神经网络, 输出层第 j 个神经元的取值计算如下:

$$F_j(X)=W_{j,:}^{\mathrm{Fa}}\tanh\left(y^{(m)}\right)+b_j^{\mathrm{F}} \tag{5}$$

注意到激活函数 $\tanh(y)$ 被两个线性函数约束, 则有 $h_{\mathrm{L},i}^{(m)}\left(y_i^{(m)}\right)\leqslant\tanh\left(y_i^{(m)}\right)\leqslant h_{\mathrm{U},i}^{(m)}\left(y_i^{(m)}\right)$, 再根据式(5)结合 $W_{j,i}^{\mathrm{Fa}}$ 的正负情况可以得到:

$$\begin{aligned}F_j(X)\leqslant F_j^{\mathrm{U},m}(X)=&\sum_{W_{j,i}^{\mathrm{Fa}}\geqslant 0}W_{j,i}^{\mathrm{Fa}}\alpha_{\mathrm{U},i}^{(m)}\left(y_i^{(m)}+\beta_{\mathrm{U},i}^{(m)}\right)\\&+\sum_{W_{j,i}^{\mathrm{Fa}}<0}W_{j,i}^{\mathrm{Fa}}\alpha_{L,i}^{(m)}\left(y_i^{(m)}+\beta_{L,i}^{(m)}\right)+b_j^{\mathrm{F}}\end{aligned} \tag{6}$$

其中，$F_j^{\mathrm{U},m}(X)$ 是 $F_j(X)$ 的上界表达式，上标 m 代表已经回溯到循环神经网络第 m 个时刻的输入层，得到了关于第 m 层输入 $X^{(m)}$ 的线性表达式。为了将 $W_{j,i}^{\mathrm{Fa}}$ 为非负与负数的情况统一起来，可以采用 $\lambda_{j,i}^{(m)}$ 和 $\Delta_{i,j}^{(m)}$ 来统一表示 $y_i^{(m)}$ 前的斜率与截距，$\lambda_{j,i}^{(m)}$ 和 $\Delta_{i,j}^{(m)}$ 的具体定义为:

$$\lambda_{j,i}^{(m)}=\begin{cases}\alpha_{\mathrm{U},i}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}\geqslant 0\\ \alpha_{\mathrm{L},i}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}<0\end{cases}\qquad \Delta_{i,j}^{(m)}=\begin{cases}\beta_{\mathrm{U},i}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}\geqslant 0\\ \beta_{\mathrm{L},i}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}<0\end{cases}$$

这样，就可以将式(6)中的两种情况合并。为了进一步简化公式表示，定义一个矩阵 $\Lambda_{j,i}^{(m)}=W_{j,i}^{\mathrm{Fa}}\lambda_{j,i}^{(m)}$。此外，将 $y_i^{(m)}=W_{i,:}^{\mathrm{aa}}a^{(m-1)}+W_{i,:}^{\mathrm{ax}}x^{(m)}+b_i^{\mathrm{a}}$ 代入，即考虑向输入层的回溯，则有:

$$\begin{aligned}F_j^{\mathrm{U},m}(X)=&\Lambda_{j,:}^{(m)}W^{\mathrm{aa}}a^{(m-1)}+\Lambda_{j,:}^{(m)}W^{\mathrm{ax}}x^{(m)}\\&+\Lambda_{j,:}^{(m)}\left(b^{\mathrm{a}}+\Delta_{:,j}^{(m)}\right)+b_j^{\mathrm{F}}\end{aligned}\tag{7}$$

其中，$\Lambda_{j,:}^{(m)}$ 代表矩阵 $\Lambda^{(m)}$ 的第 j 行，$\Delta_{:,j}^{(m)}$ 代表矩阵 $\Delta^{(m)}$ 的第 j 列，类似的记号同理。将式(7)中的系数进行整理，分别用 $\widetilde{W}_{j,r}^{\mathrm{aa}(m)}=\Lambda_{j,:}^{(m)}W_{:,r}^{\mathrm{aa}}$、$\widetilde{W}_{j,q_m}^{\mathrm{ax}(m)}=\Lambda_{j,:}^{(m)}W_{:,q_m}^{\mathrm{ax}}$ 和 $\tilde{b}_j^{(m)}=\Lambda_{j,:}^{(m)}\left(b^{\mathrm{a}}+\Delta_{:,j}^{(m)}\right)+b_j^{\mathrm{F}}$ 来表示合并后的系数，则式(7)可进一步简化为:

$$F_j^{\mathrm{U},m}(X)=\widetilde{W}_{j,:}^{\mathrm{aa}(m)}a^{(m-1)}+\widetilde{W}_{j,:}^{\mathrm{ax}(m)}x^{(m)}+\tilde{b}_j^{(m)}\tag{8}$$

至此，推导出了输出层神经元关于第 m 层输入的上界表达式。

接下来向前回溯，推导输出层神经元关于第 $m-1$ 层输入的上界表达式。注意到式(8)中 $\widetilde{W}_{j,:}^{\mathrm{aa}(m)}a^{(m-1)}$ 这一部分可回溯得到关于 $x^{(m-1)}$ 与 $a^{(m-2)}$ 的线性表达式。首先，将 $a^{(m-1)}$ 替换为 $\tanh\left(y_r^{(m-1)}\right)$，进而对其进行线性松弛，可得:

$$\begin{aligned}F_j^{\mathrm{U},m-1}(X)=&\sum_{r=1}^{s}\widetilde{W}_{j,r}^{\mathrm{aa}(m)}\lambda_{j,r}^{(m-1)}\left(y_r^{(m-1)}+\Delta_{r,j}^{(m-1)}\right)\\&+\widetilde{W}_{j,:}^{\mathrm{ax}(m)}x^{(m)}+\tilde{b}_j^{(m)}\end{aligned}\tag{9}$$

其中，$\lambda_{j,r}^{(m-1)}$ 和 $\Delta_{r,j}^{(m-1)}$ 的含义与上述 $\lambda_{j,i}^{(m)}$ 和 $\Delta_{i,j}^{(m)}$ 的含义类似，即根据 $\widetilde{W}_{j,r}^{\mathrm{aa}(m)}$ 的正负情况来判断，分别取 $\alpha_{U,r}^{(m)}$ 或 $\alpha_{L,r}^{(m)}$ 以及 $\beta_{U,r}^{(m)}$ 或 $\beta_{L,r}^{(m)}$，之后的推导同理。接下来，采用类似的方式定义矩阵 $\Lambda_{j,r}^{(m-1)}=\widetilde{W}_{j,r}^{\mathrm{aa}(m)}\lambda_{j,r}^{(m-1)}$，注意到 $y_r^{(m-1)}=W_{r,:}^{\mathrm{aa}}a^{(m-2)}+W_{r,:}^{\mathrm{ax}}x^{(m-1)}+b_r^{\mathrm{a}}$ 并将其代入式(9)。进而使用 $\widetilde{W}_{j,i}^{\mathrm{aa}(m-1)}=\Lambda_{j,:}^{(m-1)}W_{:,i}^{\mathrm{aa}}$、$\widetilde{W}_{j,q_{m-1}}^{\mathrm{ax}(m-1)}=\Lambda_{j,:}^{(m-1)}W_{:,q_{m-1}}^{\mathrm{ax}}$ 和 $\tilde{b}_j^{(m-1)}=\Lambda_{j,:}^{(m-1)}\left(b^{\mathrm{a}}+\Delta_{:,j}^{(m-1)}\right)+\tilde{b}_j^{(m)}$ 对系数进行合并整理，可得输出层关于第 $m-1$ 层输入 $x^{(m-1)}$ 的上界表达式:

$$\begin{aligned}F_j^{\mathrm{U},m-1}(X)=&\widetilde{W}_{j,:}^{\mathrm{aa}(m-1)}a^{(m-2)}+\widetilde{W}_{j,:}^{\mathrm{ax}(m-1)}x^{(m-1)}\\&+\widetilde{W}_{j,:}^{\mathrm{ax}(m)}x^{(m)}+\tilde{b}_j^{(m-1)}\end{aligned}\tag{10}$$

观察到式(10)与式(8)具有相同的形式。接下来可将 $a^{(m-2)}$ 进行替换，继续回溯得到包含关于 $x^{(m-2)}$ 的与 $a^{(m-3)}$ 线性表达式，然后重复上述迭代过程按照 $F_j(X)\leqslant F_j^{\mathrm{U},m}(X)\leqslant\cdots\leqslant F_j^{\mathrm{U},1}(X)$ 的顺序进行推导，得到输出层神经元上界关于输入的线性表达式:

$$\begin{aligned}F_j^{\mathrm{U}}(X)=F_j^{\mathrm{U},1}(X)=&\ \Lambda_{j,:}^{(0)}a^{(0)}+\sum_{k=1}^{m}\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}x^{(k)}\\&+\sum_{k=1}^{m}\Lambda_{j,:}^{(k)}\left(b^{\mathrm{a}}+\Delta_{:,j}^{(k)}\right)+b_j^{\mathrm{F}}\end{aligned}\tag{11}$$

其中，当 $k=m+1$ 时，$\Lambda_{j,:}^{(k-1)}=W_{j,:}^{\mathrm{Fa}}\odot\lambda_{j,:}^{(k-1)}$；$k\in[m]$ 时，$\Lambda_{j,:}^{(k-1)}=\left(\Lambda_{j,:}^{(k)}W^{\mathrm{aa}}\right)\odot\lambda_{j,:}^{(k-1)}$。本文用 $[m]$ 表示集合 $\{1,2,...,m\}$。

目前已得到神经元上界关于输入的表达式，但还不是具体值。为了得到具体值需要进一步将其转化为关于扰动距离 ϵ 的解析解。考虑到 $k\in[m]$。令距离度量 q 满足 $\frac{1}{p}+\frac{1}{q}=1$。根据赫尔德不等式，可得 $\sum_{k=1}^{m}\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}(x^{(k)}-x_0^{(k)})<=\sum_{k=1}^{m}\epsilon\left\|\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}\right\|_q$。结合式(11)，输出层神经元上界关于 ϵ 的解析解为:

$$\begin{aligned}\gamma_j^{\mathrm{U}}=&\Lambda_{j,:}^{(0)}a^{(0)}+\sum_{k=1}^{m}\epsilon\left\|\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}\right\|_q+\sum_{k=1}^{m}\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}x_0^{(k)}\\&+\sum_{k=1}^{m}\Lambda_{j,:}^{(k)}\left(b^{\mathrm{a}}+\Delta_{:,j}^{(k)}\right)+b_j^{\mathrm{F}}\end{aligned}\tag{12}$$

同理，第 v 个隐藏层神经元上界关于 ϵ 的解析解为:

$$\begin{aligned}u_j^{(v)}=&\Lambda_{j,:}^{(0)}a^{(0)}+\sum_{k=1}^{v}\epsilon\left\|\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}\right\|_q+\sum_{k=1}^{v}\Lambda_{j,:}^{(k)}W^{\mathrm{ax}}x_0^{(k)}\\&+\sum_{k=1}^{v}\Lambda_{j,:}^{(k)}\left(b^{\mathrm{a}}+\Delta_{:,j}^{(k)}\right)\end{aligned}\tag{13}$$

在式(13)中，边界处 $\Lambda_{j,:}^{(v-1)}$ 和 $\Lambda_{j,:}^{(v)}$ 分别取值为 $W_{j,:}^{\mathrm{aa}} \odot \lambda_{j,:}^{(v-1)}$ 和 $e_j^{\top}$。至此，已推导出循环神经网络各层神经元的近似上界。

循环神经网络输出层与第 v 个隐藏层神经元的近似下界的推导与上述过程类似，结果如下:

$$\begin{aligned}\gamma_j^{\mathrm{L}} &= \Omega_{j,:}^{(0)} a^{(0)} - \sum_{k=1}^{m} \epsilon \left\| \Omega_{j,:}^{(k)} W^{\mathrm{ax}} \right\|_q \\ &+ \sum_{k=1}^{m} \Omega_{j,:}^{(k)} W^{\mathrm{ax}} x_0^{(k)} + \sum_{k=1}^{m} \Omega_{j,:}^{(k)} \left(b^{\mathrm{a}} + \Theta_{:,j}^{(k)} \right) + b_j^{\mathrm{F}}\end{aligned} \tag{14}$$

$$\begin{aligned}l_j^{(v)} &= \Omega_{j,:}^{(0)} a^{(0)} - \sum_{k=1}^{v} \epsilon \left\| \Omega_{j,:}^{(k)} W^{\mathrm{ax}} \right\|_q \\ &+ \sum_{k=1}^{v} \Omega_{j,:}^{(k)} W^{\mathrm{ax}} x_0^{(k)} + \sum_{k=1}^{v} \Omega_{j,:}^{(k)} \left(b^{\mathrm{a}} + \Theta_{:,j}^{(k)} \right)\end{aligned} \tag{15}$$

其中，记号 $\Omega_{j,:}^{(k)}$ 和 $\Theta_{:,j}^{(k)}$ 的含义分别与 $\Lambda_{j,:}^{(k)}$ 和 $\Delta_{:,j}^{(k)}$ 对偶。

3.2 基于多路径回溯的近似区间求解

如何得到更加精确的近似区间是基于近似求解的神经网络验证方法的关键。假设对于原始样本在扰动领域范围内的所有样本经过前向传播后神经元激活前值的精确区间为 $[rl_k^{(r)}, ru_k^{(r)}]$，某种近似方法求解出的上下界构成的上近似区间为 $[l_k^{(r)}, u_k^{(r)}]$，则有 $l_k^{(r)} < rl_k^{(r)}, u_k^{(r)} > ru_k^{(r)}$。此时，$l_k^{(r)}$ 越接近 $rl_k^{(r)}$，$u_k^{(r)}$ 越接近 $ru_k^{(r)}$ 则越好。注意到不同的求解方法往往能求解出不同的上下界(上近似区间)，假设有两种求解方法求解出的区间分别为 $[l_{k,1}^{(r)}, u_{k,1}^{(r)}]$ 与 $[l_{k,2}^{(r)}, u_{k,2}^{(r)}]$，两者的交集 $[\max(l_{k,1}^{(r)}, l_{k,2}^{(r)}), \min(u_{k,1}^{(r)}, u_{k,2}^{(r)})]$ 依然是精确区间的上近似区间，并且精确程度不低于 $[l_{k,1}^{(r)}, u_{k,1}^{(r)}]$ 与 $[l_{k,2}^{(r)}, u_{k,2}^{(r)}]$。一个更紧密的区间有利于构造一个更精确的线性近似抽象，有助于提高后续神经元近似区间求解的精确程度。由此可看出，构造不同的近似区间回溯迭代过程，并利用它们的求解得到的近似区间求交集，可以得到更优的近似区间。

观察 3.1 节的回溯迭代过程和最终求得的关于输入的线性表达式可以发现，关于输入的系数是由循环神经网络的权重以及激活函数线性近似步骤得到的斜率与截距组成的。由于权重是固定的，因此对于同一个神经元的近似区间的计算而言，构造不同的回溯迭代过程需要构造不同的激活函数的线性近似方法。2.3 节以 tanh 激活函数为例介绍了经典近似求解框架使用的线性近似方式与细粒度的线性近似方式。可以观察出，细粒度近似方式较经典近似求解框架的近似方式有着不同的分类模式，可以避免经典近似求解框架中部分情况下出现的近似不紧密的现象，如图 3 所示。同时，文献[28]表明，随着待验证的神经网络模型深度越高，细粒度线性近似方式往往能够更好地改善验证的效果。循环神经网络通常具有较高的深度，因此这种方式非常适合用于求解循环神经网络验证问题中的近似区间。然而，这种细粒度的线性近似方式仍有可以改进的空间，比如 2.3 节中提及的细粒度近似方法的前两种情况，分别作切线的下边界线与上边界线与两个端点的连线平行。为了达到这种效果，需要通过二分法去找到对应的激活函数点，增大了时间开销。然而，并没有理论与实验依据表明取平行线的方式一定能取得最佳的近似效果，可构造激活函数的不同的近似方式。例如，可以选择在细粒度近似方式的前两种情况选择点 $\left(\frac{(l_k^{(r)} + u_k^{(r)})}{2}, \tanh\left(\frac{(l_k^{(r)} + u_k^{(r)})}{2} \right) \right)$ 的切线分别作为下边界线与上边界线，如图 5 所示。

基于上述分析，本文提出多路径回溯策略，即通过调整细粒度近似的情况(a)中的下边界线与情况(b)中的上边界线来构造不同的回溯路径，并计算各

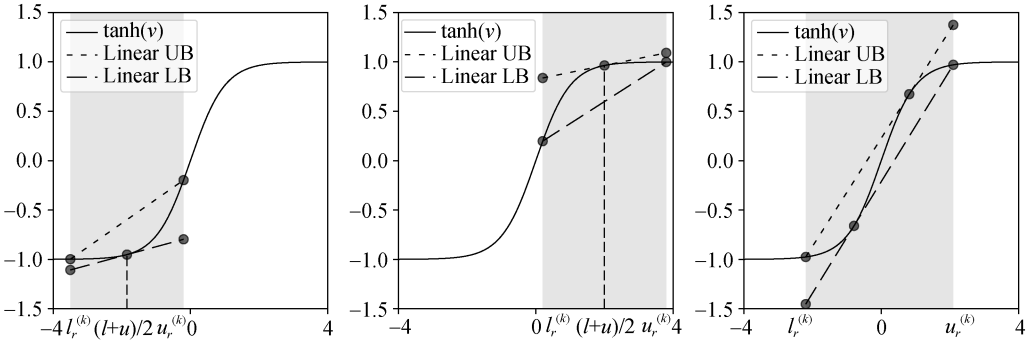

图 5 改进的细粒度近似方式

Figure 5 Improved fine-grained approximation

路径求得的近似区间的交集作为结果。具体而言, 定义一个参数 θ 来控制构造不同的细粒度近似方式。即根据参数 θ, 对于情况(a)将下边界线选取为点 $\left(l_k^{(r)}+\theta(u_k^{(r)}-l_k^{(r)}),\tanh\left(l_k^{(r)}+\theta(u_k^{(r)}-l_k^{(r)})\right)\right)$ 的切线, 对于情况(b)将上边界线选取为点 $\left(l_k^{(r)}+(1-\theta)(u_k^{(r)}-l_k^{(r)}),\tanh\left(l_k^{(r)}+(1-\theta)(u_k^{(r)}-l_k^{(r)})\right)\right)$ 的切线。每种参数 θ 的取值得到一条回溯路径, 图 5 可看作是 $\theta=0.5$ 的特例。同时, 用 n 代表回溯路径数, 选取 $\theta=0,0.1,...,\frac{n-2}{10}$ 以及细粒度近似对应的路径共计 n 条路径。由于包含了细粒度近似对应的路径, 这样可以确保比单纯使用细粒度近似方式取得的结果更精确。以 $n=7$ 为例, 选取 $\theta=0,0.1,0.2,0.3,0.4,0.5$ 与细粒度近似共计 7 条路径。将 $\theta=0,0.1,0.2,0.3,0.4,0.5$ 代表的回溯路径分别称为第 1 到第 6 条路径, 细粒度近似代表的路径称为第 7 条路径。前 6 条路径相对应于上述两种情况的激活函数线性近似方式如图 6 所示, 注意第 7 条路径的情况已在图 3 中给出。接下来, 定义不同路径下的近似区间上下界记号。令 $\gamma_{j,k}^{\mathrm{U}}$ 和 $\gamma_{j,k}^{\mathrm{L}}$ 分别表示第 k 条路径求解出的输出层第 j 个神经元的上下界, $u_{j,k}^{(z)}$ 和 $l_{j,k}^{(z)}$ 分别表示第 k 条路径求解出的第 z 层隐藏层第 j 个神经元的上下界。以 $\gamma_{j,k}^{\mathrm{U}}$ 为例, 根据式(12)可得:

$$\begin{aligned}\gamma_{j,k}^{\mathrm{U}}&=\Lambda_{j,:,k}^{(1)}a^{(0)}+\sum_{z=1}^{m}\epsilon\left\|\Lambda_{j,:,k}^{(z)}W^{\mathrm{ax}}\right\|_q\\&+\sum_{z=1}^{m}\Lambda_{j,:,k}^{(z)}W^{\mathrm{ax}}x_0^{(z)}+\sum_{z=1}^{m}\Lambda_{j,:,k}^{(z)}\left(b^{\mathrm{a}}+\Delta_{:,j,k}^{(z)}\right)+b_j^{\mathrm{F}}\end{aligned}\tag{16}$$

式(16)中部分符号具体定义如下:

$$\Lambda_{j,:,k}^{(z-1)}=\begin{cases}W_{j,:}^{\mathrm{Fa}}\odot\lambda_{j,:,k}^{(z-1)} & \text{if } z=m+1\\ \left(\Lambda_{j,:,k}^{(z)}W^{\mathrm{aa}}\right)\odot\lambda_{j,:,k}^{(z-1)} & \text{if } z\in[m]\end{cases}$$

$$\lambda_{j,i,k}^{(m)}=\begin{cases}\alpha_{\mathrm{U},i,k}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}\geqslant 0\\ \alpha_{\mathrm{L},i,k}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}<0\end{cases}$$

$$\Delta_{i,j,k}^{(m)}=\begin{cases}\beta_{\mathrm{U},i,k}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}\geqslant 0\\ \beta_{\mathrm{L},i,k}^{(m)} & \text{if } W_{j,i}^{\mathrm{Fa}}<0\end{cases}$$

对于 $\forall r\in s,z\in[m-1]$ 则有:

$$\lambda_{j,r,k}^{(z)}=\begin{cases}\alpha_{\mathrm{U},r,k}^{(z)} & \text{if } \Lambda_{j,:,k}^{(z+1)}W_{:,r}^{\mathrm{aa}}\geqslant 0\\ \alpha_{\mathrm{L},r,k}^{(z)} & \text{if } \Lambda_{j,:,k}^{(k+1)}W_{:,r}^{\mathrm{aa}}<0\end{cases}$$

$$\Delta_{r,j,k}^{(z)}=\begin{cases}\beta_{\mathrm{U},r,k}^{(z)} & \text{if } \Lambda_{j,:,k}^{(z+1)}W_{:,r}^{\mathrm{aa}}\geqslant 0\\ \beta_{\mathrm{L},r,k}^{(z)} & \text{if } \Lambda_{j,:,k}^{(z+1)}W_{:,r}^{\mathrm{aa}}<0\end{cases}$$

而 $\gamma_{j,k}^{\mathrm{L}}$、$u_{j,k}^{(z)}$ 和 $l_{j,k}^{(z)}$ 可分别根据类似的过程求得, 其中 $\alpha_{\mathrm{U},r,k}^{(z)}$ 和 $\beta_{\mathrm{U},r,k}^{(z)}$ 根据 $u_{j,k}^{(z)}$ 和 $l_{j,k}^{(z)}$ 结合对应路径的激活函数近似方式得到, 这里不再赘述。

基于上述多路径回溯策略的循环神经网络近似区间求解如算法 1 所示。其中, $l_{j,k}^{(i)}$ 和 $u_{j,k}^{(i)}$ 分别对应第 k 条路径得到的区间上下界, $[l_j^{(i)},u_j^{(i)}]$ 代表 n 条路径求出的近似区间的交集, 也是该方法最终实际求得的第 i 层第 j 个神经元的近似区间。

算法 1. 基于多路径回溯的近似区间求解

输入: 循环神经网络的权重与偏置参数, 扰动范围 ϵ, 距离度量 p, 回溯路径数 n, 原始样本 X_0;

输出: 循环层各神经元激活前值近似区间的上下界, 输出层神经元近似区间的上下界。

1. FOR $i\leftarrow$1 TO m DO
2. FOR $j\leftarrow$1 TO s DO
3. FOR $k\leftarrow$1 TO n DO
4. $l_{j,k}^{(i)},u_{j,k}^{(i)}\leftarrow$根据回溯迭代公式计算上下界
5. $[l_j^{(i)},u_j^{(i)}]\leftarrow\bigcap_{k=1}^{n}[l_{j,k}^{(i)},u_{j,k}^{(i)}]$ //将各个路径近似区间取交集得到更精确的近似区间
6. FOR $k\leftarrow$1 TO n DO
7. $\alpha_{\mathrm{L},j,k}^{(i)},\alpha_{\mathrm{U},j,k}^{(i)},\beta_{\mathrm{L},j,k}^{(i)},\beta_{\mathrm{U},j,k}^{(i)}\leftarrow$交集区间 $[l_j^{(i)},u_j^{(i)}]$ 在不同的激活函数线性近似方式计算得到
8. FOR $j\leftarrow$1 TO s DO
9. FOR $k\leftarrow$1 TO n DO
10. $\gamma_{j,k}^{\mathrm{U}},\gamma_{j,k}^{\mathrm{L}}\leftarrow$根据回溯迭代公式计算上下界
11. $[\gamma_j^{\mathrm{U}},\gamma_j^{\mathrm{L}}]\leftarrow\bigcap_{k=1}^{n}[\gamma_{j,k}^{\mathrm{U}},\gamma_{j,k}^{\mathrm{L}}]$

算法 1 保持了经典近似求解框架的高效性。设 m 表示网络层数, s 表示隐藏层神经元数目。根据文献[20]可知, 采用单路径回溯的经典近似求解方法的时间复杂度为 $O\left(m^2s^3\right)$。因此, 算法 1 的时间复杂度为 $O\left(nm^2s^3\right)$, 其中 n 为多路径回溯策略中的路径数。当 n 选取一个较小的整数, 例如 n=7 时, 算法 1 的耗

时与经典近似求解方法规模相同, 相较于有着指数时间复杂度的基于优化的精确求解方法依然有着明显的优势。4.2 节将通过实验对路径数 n 的选取进行分析。

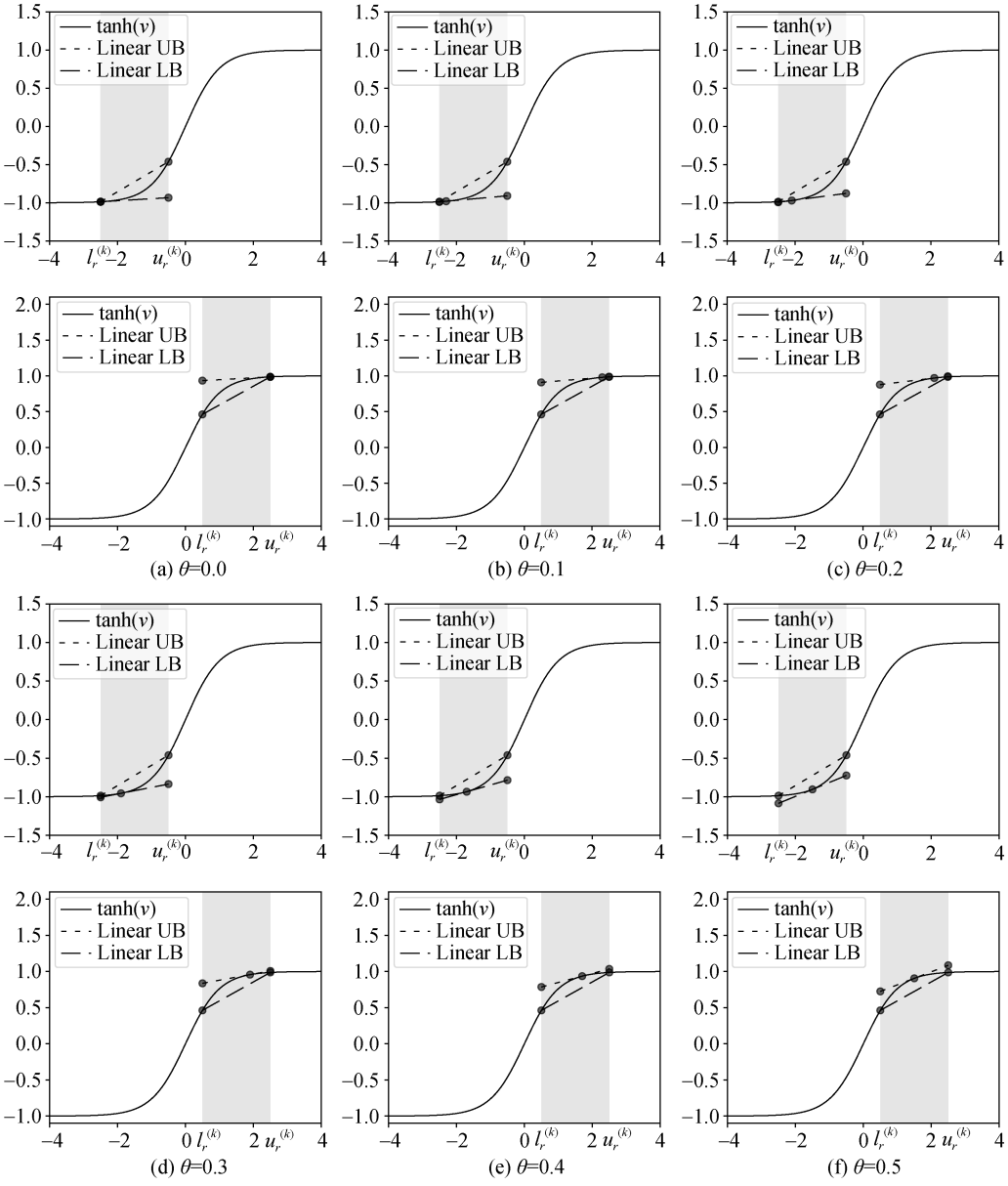

图 6　n=7 时前 6 条路径使用的激活函数近似方式

Figure 6　Approximation of the activation function used by the first 6 paths with n=7

3.3　健壮半径求解

在循环神经网络近似区间求解的基础上, 进一步对健壮半径进行求解。一种可行的二分法思想是: 预设一个扰动距离的初始值 ϵ, 由此求得输出层神经元近似区间上下界的具体值; 通过比较输出层目标神经元近似区间的下界与其他各神经元近似区间的上界来判断循环神经网络是否具有健壮性; 并根据判断结果调整 ϵ 的取值, 进入下一轮的计算, 直至 ϵ 值达到规定的精度。设 label(X) 为 j, 则每一轮计算会检查 $\forall i \neq j\ \gamma_j^{\mathrm{L}} > \gamma_i^{\mathrm{U}}$ 是否满足。若满足则可判定循环神经网络健壮, 进而对 ϵ 进行放大; 反之, 则无法判定网络是否健壮, 进而对 ϵ 进行缩小。然而, 这种经典的二分处理方式存在一个问题。在每一轮计算中, 上下界 γ_i^{U} 和 γ_j^{L} 各自都经过了一次赫尔德不等式的放缩, 造成了一定程度的精度缺失, 可能导致一些本可被判定为健壮的情况被判断为不确定。

针对这一问题, VR-RRS 采用一种改进的二分法来求解健壮半径。不直接计算上下界 γ_i^{U} 和 γ_j^{L}, 而是定义一个函数 $D_i(X) = F_j(X) - F_i(X)$。根据 3.1 节的

推导可得:

$$D_i(X) \geqslant D_i^{\mathrm{L}}(X) = \Omega_{j,:}^{(0)} a^{(0)} + \sum_{k=1}^{m} \Omega_{j,:}^{(k)} W^{\mathrm{ax}} x^{(k)} + \sum_{k=1}^{m} \Omega_{j,:}^{(k)} \left(b^{\mathrm{a}} + \Theta_{:,j}^{(k)} \right) + b_j^{\mathrm{F}}$$

将 $D_i(X)$ 的近似区间下界关于 ϵ 的解析记作 δ_i^{L}, 则有:

$$\delta_i^{\mathrm{L}} = \Omega_{i,:}^{(0)} a^{(0)} - \sum_{k=1}^{m} \epsilon \left\| \Omega_{i,:}^{(k)} W^{\mathrm{ax}} \right\|_q + \sum_{k=1}^{m} \Omega_{i,:}^{(k)} W^{\mathrm{ax}} x_0^{(k)} + \sum_{k=1}^{m} \Omega_{i,:}^{(k)} \left(b^{\mathrm{a}} + \Theta_{:,i}^{(k)} \right) + b_i^{\mathrm{F}} \tag{17}$$

其中部分符号定义如下:

$$\Omega_{i,:}^{(k-1)} = \begin{cases} (W_{j,:}^{\mathrm{Fa}} - W_{i,:}^{\mathrm{Fa}}) \odot \omega_{i,:}^{(k-1)} & \text{if} \quad k = m+1 \\ \left(\Omega_{i,:}^{(k)} W^{\mathrm{aa}} \right) \odot \omega_{i,:}^{(k-1)} & \text{if} \quad k \in [m] \end{cases}$$

通过判断 δ_i^{L} 是否大于 0 即可得出神经网络是否具有健壮性的结论。这样减小了因不等式放缩带来的对于近似区间精确度的影响。式(17)仅展示了单一路径下 δ_i^{L} 的求解, 在实际求解过程中使用的是 3.2 节提出的多路径回溯策略。

综上所述, 基于改进的二分法的健壮半径求解如算法 2 所示。

算法 2. 基于改进的二分法的健壮半径求解

输入: 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}, 二分法终止精度 acc;

输出: 健壮半径 ϵ_{cert}。

1. $\epsilon_{\mathrm{ub}} = \epsilon_{\mathrm{ini}}, \epsilon_{\mathrm{lb}} = 0$
2. WHILE $\forall i \neq j, \delta_i^{\mathrm{L}} > 0$ DO
3. $\epsilon_{\mathrm{lb}} \leftarrow \epsilon_{\mathrm{ub}}, \epsilon_{\mathrm{ub}} \leftarrow 2 \times \epsilon_{\mathrm{lb}}$
4. FOR $i \leftarrow 1$ TO t DO
5. IF i != j
6. $\delta_i^{\mathrm{L}} \leftarrow$代入 ϵ_{ub} 计算得到
7. WHILE $\epsilon_{\mathrm{ub}} - \epsilon_{\mathrm{lb}} > \mathrm{acc}$ DO
8. $\epsilon \leftarrow \dfrac{(\epsilon_{\mathrm{lb}} + \epsilon_{\mathrm{ub}})}{2}$
9. $\delta_j^{\mathrm{L}} \leftarrow$代入 ϵ 计算得到 δ_j^{L}
10. FOR $i \leftarrow 1$ TO t DO
11. IF i != j
12. $\delta_i^{\mathrm{L}} \leftarrow$代入 ϵ 计算得到
13. IF $\forall i \neq j, \delta_i^{\mathrm{L}} > 0$ then
14. $\epsilon_{\mathrm{lb}} \leftarrow \epsilon$
15. ELSE
16. $\epsilon_{\mathrm{ub}} \leftarrow \epsilon$
17. RETURN $\epsilon_{\mathrm{cert}} = \epsilon_{\mathrm{lb}}$

在算法 2 中, 第 2-6 行为确定二分法下界与上界的过程, 即确定一个满足健壮性判定条件的扰动下界 ϵ_{lb}, 以及一个不满足健壮性判定条件的扰动上界 ϵ_{ub}。第 7-16 行为采用改进的二分法确定健壮半径的过程, 当上下界之差小于预设精度后即返回二分下界作为最终求得的健壮半径 ϵ_{cert}。

4 实验与分析

本节从求解出的健壮半径与健壮性验证成功率这两个角度通过实验对 VR-RRS 进行评估, 并与已有的循环神经网络验证方法 POPQORN[23]进行对比分析。同时, 为了证明本方法中多路径回溯策略的有效性, 对使用多路径回溯与仅使用单路径回溯策略的实验结果进行对比分析。实验选择在 64 位 Ubuntu18.04 平台上进行, CPU 为 16 核的 Intel Core i9-9900K, 频率为 3.60GHZ, 内存为 32GB。各循环神经网络均使用 Python 3.8 来实现。

关于训练循环神经网络数据集, 实验选择手写数字数据集 MNIST。它是神经网络验证大赛 VNN-COMP 采用的三种数据集之一。另两种分别为彩色图像分类数据集 CIFAR 和无人机防撞系统数据集 ACAS Xu。然而, 由于循环神经网络识别彩色图像准确率较低, 且 ACAS Xu 的样本维度过低, 这两种数据集不太适用于循环神经网络的建模处理。

MNIST 数据集由手写数字 0-9 的图片样本组成。每张图片为 28×28 像素的灰度图像, 其中每个像素点取值范围为[0, 255], 样本示例如图 7 所示。为了适应循环神经网络的特点, 实验对样本进行一定的调整。例如, 对于一个 4 层的循环神经网络, 样本格式被调整为 4×196, 即每个时刻输入层大小为 196。所有训练样本都被标准化为均值为 0, 标准差为 1 的数据。

图 7 MNIST 数据集样本示例

Figure 7 Example samples of the MNIST dataset

表 1 VR-RRS 与 POPQORN 求解出的健壮半径对比

Table 1 Comparison of robust radius calculated by VR-RRS and POPQORN

神经网络模型	距离度量	POPQORN (均值/最大值/最小值)	VR-RRS (均值/最大值/最小值)	均值改善率(%)
RNN-4-32	l_1	0.8811 / 1.6348 / 0.2656	1.0696 / 2.1508 / 0.2917	21.4
	l_2	0.1836 / 0.3433 / 0.0553	0.2221 / 0.4148 / 0.0616	21.0
	l_∞	0.0172 / 0.0320 / 0.0052	0.0208 / 0.0387 / 0.0056	21.0
RNN-4-64	l_1	1.0994 / 1.8945 / 0.6284	1.3116 / 2.5253 / 0.6757	19.3
	l_2	0.2219 / 0.3821 / 0.1260	0.2647 / 0.4971 / 0.1320	19.3
	l_∞	0.0205 / 0.0355 / 0.0117	0.0245 / 0.0458 / 0.0124	19.6
RNN-7-32	l_1	0.3347 / 0.5811 / 0.0171	0.4197 / 0.7351 / 0.0183	25.4
	l_2	0.0905 / 0.1490 / 0.0043	0.1137 / 0.2041 / 0.0043	25.6
	l_∞	0.0113 / 0.0185 / 0.0005	0.0141 / 0.0240 / 0.0006	25.1
RNN-7-64	l_1	0.4433 / 0.7646 / 0.2617	0.5417 / 0.9187 / 0.2966	22.2
	l_2	0.1153 / 0.1998 / 0.0696	0.1406 / 0.2430 / 0.0767	21.9
	l_∞	0.0141 / 0.0245 / 0.0086	0.0172 / 0.0294 / 0.0094	22.1
RNN-14-32	l_1	0.1384 / 0.2744 / 0.0476	0.1906 / 0.3847 / 0.0595	37.7
	l_2	0.0483 / 0.0935 / 0.0169	0.0664 / 0.1308 / 0.0216	37.4
	l_∞	0.0085 / 0.0163 / 0.0029	0.0116 / 0.0229 / 0.0038	36.5
RNN-14-64	l_1	0.1521 / 0.2505 / 0.0847	0.1964 / 0.3188 / 0.1075	29.1
	l_2	0.0510 / 0.0840 / 0.0277	0.0659 / 0.1174 / 0.0361	29.2
	l_∞	0.0088 / 0.0145 / 0.0047	0.0114 / 0.0195 / 0.0061	29.1

4.1 方法性能比较

(1) 健壮半径

为了能够全面准确地分析对比 VR-RRS 与 POPQORN, 本文在多个循环神经网络模型上进行健壮半径求解的实验。这些 RNN 模型具有不同的层数与隐藏层尺寸。为了叙述的方便, 使用 RNN-4-32 代表一个层数为 4, 隐藏层尺寸为 32 的循环神经网络模型, 其余模型也用相同的方式表示。同时, 在每个模型上的实验均使用在测试集中能够被该模型正确分类的 100 个样本。VR-RRS 中的回溯路径数设置为 7, 即回溯路径为 $\theta = 0, 0.1, 0.2, 0.3, 0.4, 0.5$ 以及细粒度近似代表的路径。表 1 给出了两种方法在不同循环神经网络模型和距离度量下求得的健壮半径详细结果, 包括均值、最大值、最小值。

从实验结果可以看出, 在所有神经网络模型以及距离度量下, VR-RRS 求解出的健壮半径均值相较于 POPQORN 提高了 19.3%~37.7%不等, 并且健壮半径的最大值和最小值也有着显著的提升。较大的健壮半径意味着更高的健壮性保障, 这充分说明了 VR-RRS 具备良好的性能。此外, 从表 1 可以看出, 对于层数较高的循环神经网络模型, VR-RRS 相较于 POPQORN 求解出的健壮半径的改善率也相对较高。对于 4 层的网络改善率为 19.3%~21.4%, 对于 7 层的网络改善率为 21.9%~25.6%, 而对于 14 层的网络改善率为 29.1%~37.7%。这一结果说明本方法能够有效地验证具有较大深度的循环神经网络。

(2) 验证成功率

基于近似求解的神经网络验证方法是单边性的保障手段。当方法给出模型健壮的结论, 如扰动不超过健壮半径时, 那么该结论一定是可靠的, 此时验证成功。反之, 方法会给出不确定的结论, 但无法断定模型一定不健壮, 此时可以看做验证不成功。这也是制约近似求解验证方法应用的主要因素。因此, 提高这类方法的成功率显得尤为重要。表 2 给出了 VR-RRS 与 POPQORN 在 RNN-7-32 与 RNN-7-64 模型上对于不同大小的扰动(基于 l_1 距离度量)下的验证成功率。

从实验结果可以看出, 在不同大小的扰动情况下, VR-RRS 的验证成功率较 POPQORN 均有较大提升。其中, 对于 RNN-7-32 模型而言, 当扰动大于 0.6 时, POPQORN 已不能够进行成功的验证, 而 VR-RRS 依然能够验证少部分样本; 对于 RNN-7-64, 在扰动值等于 0.6 时, POPQORN 仅能验证出 100 张实验样本中的 4 张, 而 VR-RRS 能验证 26 张, 成功率是 POPQORN 的 6.5 倍, 显示了 VR-RRS 较为良好的可用性。

表 2 VR-RRS 与 POPQORN 验证成功率对比(验证样本数为 100)

Table 2 Comparison of verification success rate of VR-RRS and POPQORN (the number of verification samples is 100)

神经网络模型	方法	在不同扰动下的验证成功率(%)				
		扰动 0.3	扰动 0.4	扰动 0.5	扰动 0.6	扰动 0.7
RNN-7-64	VR-RRS	99	87	60	26	8
	POPQORN	93	69	26	4	2
RNN-7-32	VR-RRS	80	55	20	5	1
	POPQORN	63	23	4	0	0

以上用两种方法在不同扰动类型以及不同网络模型进行对比的实验结果表明, VR-RRS 的验证精度较 POPQORN 有显著提升。

本文的工作与同类工作 POPQORN[23]均受到经典近似求解框架的启发。与 POPQORN 直接沿用了经典近似求解框架中激活函数的近似方法不同的是, 本文提出了一种针对非线性激活函数的多路径回溯求解策略以提升求解的精度。此外, 在健壮半径求解阶段, 本文改进了 POPQORN 所使用的二分法求解方式。本文提出的多路径回溯策略在单个路径的构造上借鉴了文献[28]提出的细粒度的近似方式, 并在此基础上做了细节上的改进。同时, 与文献[28]旨在验证卷积神经网络不同的是, 本文验证的对象为循环神经网络。

值得指出的是, 另一项研究循环神经网络健壮性验证的工作 CERT-RNN[24]亦取得了较优的性能。该工作基于使用 zonotope 抽象域的验证框架 DeepZ[29], 并在此基础上改进了 tanh 激活函数的抽象变换方法。这一框架专门支持 l_∞ 范数度量下健壮半径的高效求解。与该工作不同的是, 本文的工作 VR-RRS 基于经典近似求解框架, 并在此基础上提出了一种多路径回溯的近似区间求解策略, 支持在各种距离度量下有效地求解健壮半径, 如 l_1、l_2、l_∞ 范数等。

4.2 多回溯路径对性能的影响分析

使用多路径回溯策略进行近似区间求解是本文提出的方法 VR-RRS 较经典近似求解框架的一个显著的改进, 也是得以提升近似区间求解精度与健壮性验证性能的关键。因此, 对使用多路径回溯策略与只使用单路径回溯的朴素方法的性能进行对比分析是有意义的。首先, 本文在 RNN-14-32 网络模型上使用多路径回溯策略(回溯路径数 $n=7$)以及构成该多路径的各个单路径回溯策略进行了健壮半径求解实验, 数据结果如表 3 所示。

从表 3 可以观察到, 在 7 种单路径回溯策略中, $\theta=0.4$ 的策略求解出的健壮半径最大, $\theta=0$ 的策略求解出的健壮半径最小。原始细粒度近似方法并未求解出最大的健壮半径, 这也证实了本工作在 2.2 节的分析猜想。同时, 即使相较于 $\theta=0.4$ 的策略求解出的健壮半径结果, 使用多路径回溯策略仍提升了约 15%, 这进一步证明了多路径回溯策略的有效性。

从以上实验可以看出多路径回溯策略对于求解效果的改善, 然而并非路径越多越好。如果路径选择不当, 求解出的近似区间的精确性很低, 对于多路径回溯策略不仅没有帮助而且增大了求解的时间开销。另一方面, 基于近似求解的方法也存在着精度上的理论上限[30]。因此, 有必要探究路径数量对求解效果的影响。本文在 RNN-14-32 网络模型上选取不同的路径数开展了健壮半径求解实验, 数据结果如表 4 所示。

从表 4 可以观察到, 在路径数 n 由 5 增长至 7 的过程中, 多路径回溯策略求得的健壮半径有着一定的增幅。另一方面, 当路径数 n 大于 7 时得到的结果与 $n=7$ 几乎相同。平衡考虑时间与性能, 本文认为

表 3 使用 7 种单路径回溯策略与多路径回溯策略求解出的健壮半径对比

Table 3 Comparison of robust radius calculated by 7 single-path backtracking strategies and the multi-path backtracking strategy

距离度量	$\theta=0$	$\theta=0.1$	$\theta=0.2$	$\theta=0.3$	$\theta=0.4$	$\theta=0.5$	细粒度	多路径
l_1	0.1372	0.1445	0.1576	0.1597	0.1647	0.1628	0.1613	0.1906
l_2	0.0471	0.0499	0.0532	0.0549	0.0574	0.0567	0.0552	0.0664
l_∞	0.0080	0.0088	0.0097	0.098	0.0101	0.0099	0.0098	0.0116

表 4 选取不同路径数 n 的多路径回溯策略求解出的健壮半径对比

Table 4 Comparison of robust radius calculated by the multi-path backtracking strategy with different path number n

距离度量	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$
l_1	0.1800	0.1868	0.1906	0.1908	0.1908
l_2	0.0627	0.0650	0.0664	0.0664	0.0664
l_∞	0.0111	0.0114	0.0116	0.0116	0.0116

$n=7$ 是相对最为合理的选择。在前一实验中观察到 $\theta=0.4$ 或 $\theta=0.5$ 的单路径回溯策略求解出的结果最佳, 这在一定程度上也可以作为选择路径的依据。

根据 3.2 节的分析, 采用多路径回溯策略的时间复杂度原则上是单路径回溯的 n 倍, 其中 n 为路径数。然而, 值得注意的是, 对于循环神经网络同一层的同一个神经元而言, 不同路径的近似区间计算过程互不影响, 因此可以采用多线程技术进行并行计算, 这将大大减少计算时间。本文在 RNN-14-32 网络模型上选取不同路径数, 并分别采用单线程、多线程方法开展了时间对比实验, 实验结果如表 5 所示。

表 5 选取不同路径数 n 的多路径回溯策略的求解时间对比

Table 5 Comparison of calculating time by the multi-path backtracking strategy with different path number n

方法	不同路径数的求解时间(秒)				
	$n=1$	$n=4$	$n=5$	$n=6$	$n=7$
单线程	45.7	198.8	254.1	304.6	362.0
多线程	45.7	52.6	69.3	89.2	121.9

从表 5 可以观察到, 在使用单线程进行计算的情况下, 求解时间与路径数呈线性相关。使用了多线程进行计算后, 选取最优路径数 $n=7$ 的计算时间为 121.9 秒, 是使用单路径求解所需时间的 2.6 倍。同类方法 POPQORN 采用的是单路径求解方式, 求解时间与 $n=1$ 的情形相当。可以看出, 本文提出的方法较大幅地提升了验证精度, 而引入的额外时间代价在可接受的范围内, 且相较于使用优化技术的精确求解方法仍有较大优势。

5 总结

本文提出了基于健壮半径求解的循环神经网络验证方法 VR-RRS。该方法采用近似求解的方式, 针对经典近似求解框架精度较低的问题, 面向循环神经网络使用的非线性激活函数设计了一种多路径回溯的近似区间求解策略, 并在此基础上采用改进的二分法对健壮半径进行求解。实验结果表明 VR-RRS 较已有性能最佳的同类方法 POPQORN 能够求解出更优的健壮半径, 并拥有更高的验证成功率, 而仅需要较小的额外时间开销。本方法目前仅面向基础类型的循环神经网络。后续的工作计划对 LSTM、GRU 等扩展类型的循环神经网络进行验证。这两类循环神经网络的结构较之于基础类型的网络更为复杂, 例如包含非线性门的耦合。如何针对这样的结构开发出可行且精确的近似区间和健壮半径求解方法是研究的一大挑战。

参考文献

[1] LeCun Y, Bottou L, Bengio Y, et al. Gradient-Based Learning Applied to Document Recognition[J]. *Proceedings of the IEEE*, 1998, 86(11): 2278-2324.

[2] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J]. *Neural Computation*, 1997, 9(8): 1735-1780.

[3] Elman J L. Finding Structure in Time[J]. *Cognitive Science*, 1990, 14(2): 179-211.

[4] He K M, Zhang X Y, Ren S Q, et al. Deep Residual Learning for Image Recognition[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 770-778.

[5] Huang Z H, Xu W, Yu K. Bidirectional LSTM-CRF Models for Sequence Tagging[EB/OL]. 2015: arXiv: 1508.01991. https://arxiv.org/abs/1508.01991

[6] Zhou P, Shi W, Tian J, et al. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification[C]. *The 54th Annual Meeting of the Association for Computational Linguistics* (*Volume 2*: *Short Papers*), 2016: 207-212.

[7] Dong Y P, Su H, Zhu J. Interpretability Analysis of Deep Neural Networks with Adversarial Examples[J]. *Acta Automatica Sinica*, 2022, 48(1): 75-86.
(董胤蓬, 苏航, 朱军. 面向对抗样本的深度神经网络可解释性分析[J]. *自动化学报*, 2022, 48(1): 75-86.)

[8] Goodfellow I J, Shlens J, Szegedy C. Explaining and Harnessing Adversarial Examples[EB/OL]. 2014: arXiv: 1412.6572. https://arxiv.org/abs/1412.6572

[9] Moosavi-Dezfooli S M, Fawzi A, Frossard P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition* , 2016: 2574-2582.

[10] Xiao C W, Li B, Zhu J Y, et al. Generating Adversarial Examples with Adversarial Networks[C]. *The 27th International Joint Conference on Artificial Intelligence*, 2018: 3905-3911.

[11] Katz G, Barrett C, Dill D L, et al. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks[C]. *International Conference on Computer Aided Verification*, 2017: 97-117.

[12] Katz G, Huang D A, Ibeling D, et al. The Marabou Framework for Verification and Analysis of Deep Neural Networks[C]. *International Conference on Computer Aided Verification*, 2019: 443-452.

[13] Lomuscio A, Maganti L. An Approach to Reachability Analysis for

Feed-Forward ReLU Neural Networks[EB/OL]. 2017: arXiv: 1706.07351. https://arxiv.org/abs/1706.07351

[14] Botoeva E, Kouvaros P, Kronqvist J, et al. Efficient Verification of ReLU-Based Neural Networks via Dependency Analysis[J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, 34(4): 3291-3299.

[15] Tjeng V, Xiao K, Tedrake R. Evaluating Robustness of Neural Networks with Mixed Integer Programming[EB/OL]. 2017: arXiv: 1711.07356. https://arxiv.org/abs/1711.07356

[16] Dutta S, Jha S, Sankaranarayanan S, et al. Output Range Analysis for Deep Feedforward Neural Networks[C]. *NASA Formal Methods Symposium*, 2018: 121-138.

[17] Raghunathan A, Steinhardt J, Liang P. Certified Defenses Against Adversarial Examples[EB/OL]. 2018: arXiv: 1801.09344. https://arxiv.org/abs/1801.09344

[18] Wang S Q, Pei K X, Whitehouse J, et al. Efficient Formal Safety Analysis of Neural Networks[C]. *The 32nd International Conference on Neural Information Processing Systems*, 2018: 6369-6379.

[19] Weng T W, Zhang H, Chen H G, et al. Towards Fast Computation of Certified Robustness for ReLU Networks[EB/OL]. 2018: arXiv: 1804.09699. https://arxiv.org/abs/1804.09699

[20] Zhang H, Weng T W, Chen P Y, et al. Efficient Neural Network Robustness Certification with General Activation Functions[C]. *The 32nd International Conference on Neural Information Processing Systems*, 2018: 4944-4953.

[21] Singh G, Gehr T, Püschel M, et al. An Abstract Domain for Certifying Neural Networks[J]. *Proceedings of the ACM on Programming Languages*, 2019, 3(POPL): 1-30.

[22] Boopathy A, Weng T W, Chen P Y, et al. CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks[C]. *The Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019: 3240-3247.

[23] Ko C Y, Lyu Z Y, Weng T W, et al. POPQORN: Quantifying Robustness of Recurrent Neural Networks[EB/OL]. 2019: arXiv: 1905.07387. https://arxiv.org/abs/1905.07387

[24] Du T Y, Ji S L, Shen L J, et al. Cert-RNN: Towards Certifying the Robustness of Recurrent Neural Networks[C]. *The 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021: 516-534.

[25] Cho K, van Merrienboer B, Gulcehre C, et al. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation[C]. *The 2014 Conference on Empirical Methods in Natural Language Processing*, 2014: 1724-1734.

[26] Papernot N, McDaniel P, Swami A, et al. Crafting Adversarial Input Sequences for Recurrent Neural Networks[C]. *MILCOM 2016 - 2016 IEEE Military Communications Conference*, 2016: 49-54.

[27] Jia R, Liang P. Adversarial Examples for Evaluating Reading Comprehension Systems[C]. *The 2017 Conference on Empirical Methods in Natural Language Processing*, 2017: 2021-2031.

[28] Wu Y T, Zhang M. Tightening Robustness Verification of Convolutional Neural Networks with Fine-Grained Linear Approximation[J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, 35(13): 11674-11681.

[29] Singh G, Gehr T, Mirman M, et al. Fast and Effective Robustness Certification[C]. *The 32nd International Conference on Neural Information Processing Systems*, 2018: 10825-10836.

[30] Salman H, Yang G, Zhang H, et al. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks[C]. *The 33rd International Conference on Neural Information Processing Systems*, 2019: 9832-9842.

赵亮 于 2012 年在比萨大学计算机科学专业获得博士学位。现任西安电子科技大学计算机科学与技术学院副教授。研究领域为形式化验证、时序逻辑。研究领域为神经网络建模与验证、形式化验证、时序逻辑程序设计。Email: lzhao@xidian.edu.cn

戚润川 于 2020 年在西安电子科技大学计算机科学与技术专业获得学士学位。现在西安电子科技大学计算机科学与技术专业攻读硕士学位。研究领域为神经网络的鲁棒性验证。Email: rcqi@stu.xidian.edu.cn

段鑫民 于 2021 年在合肥工业大学信息与计算科学专业获得学士学位。现在西安电子科技大学计算机科学与技术专业攻读硕士学位。研究领域为神经网络的形式化验证。Email: 21031211458@stu.xidian.edu.cn

李春奕 于 2018 年在中南大学计算机科学与技术专业获得学士学位。现在西安电子科技大学计算机科学与技术专业攻读博士学位。研究领域为形式化验证、需求工程、神经网络验证。Email: cyli_322@stu.xidian.edu.cn

王小兵 于 2009 年在西安电子科技大学大学计算机应用技术专业获得博士学位。现任西安电子科技大学计算机科学与技术学院副教授。研究领域为高可信软件技术、形式化验证。Email: xbwang@mail.xidian.edu.cn