软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(7):2464−2481 [doi: 10.13328/j.cnki.jos.006585]

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

基于多路径回溯的神经网络验证方法*

郑 烨, 施晓牧, 刘嘉祥

(深圳大学 计算机与软件学院, 广东 深圳 518060)

通信作者: 刘嘉祥, E-mail: jiaxiang0924@gmail.com

摘 要: 基于线性抽象的符号传播方法在神经网络验证中具有重要地位. 针对这类方法, 提出了多路径回溯的概念. 现有方法可看作仅使用单条回溯路径计算每个神经网络节点的上下界, 是这一概念的特例. 使用多条回溯路径, 可以有效地改善这类方法的精度. 在数据集 ACAS Xu, MNIST 和 CIFAR10 上, 将多路径回溯方法与使用单条回溯路径的 DeepPoly 进行定量比较, 结果表明, 多路径回溯方法能够获得明显的精度提升, 而仅引入较小的额外时间代价. 此外, 在数据集 MNIST 上, 将多路径回溯方法与使用全局优化的 Optimized LiRPA 比较, 结果表明, 该方法仍然具有精度优势.

关键词: 神经网络验证; 符号传播; 抽象解释; 多路径回溯

中图法分类号: TP311

中文引用格式: 郑烨, 施晓牧, 刘嘉祥. 基于多路径回溯的神经网络验证方法. 软件学报, 2022, 33(7): 2464–2481. http://www.jos.org.cn/1000-9825/6585.htm

英文引用格式: Zheng Y, Shi XM, Liu JX. Multi-path Back-propagation Method for Neural Network Verification. Ruan Jian Xue Bao/Journal of Software, 2022, 33(7): 2464–2481 (in Chinese). http://www.jos.org.cn/1000-9825/6585.htm

Multi-path Back-propagation Method for Neural Network Verification

ZHENG Ye, SHI Xiao-Mu, LIU Jia-Xiang

(College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

Abstract: Symbolic propagation methods based on linear abstraction play a significant role in neural network verification. This study proposes the notion of multi-path back-propagation for these methods. Existing methods are viewed as using only a single back-propagation path to calculate the upper and lower bounds of each node in a given neural network, being specific instances of the proposed notion. Leveraging multiple back-propagation paths effectively improves the accuracy of this kind of method. For evaluation, the proposed method is quantitatively compared using multiple back-propagation paths with the state-of-the-art tool DeepPoly on benchmarks ACAS Xu, MNIST, and CIFAR10. The experiment results show that the proposed method achieves significant accuracy improvement while introducing only a low extra time cost. In addition, the multi-path back-propagation method is compared with the Optimized LiRPA based on global optimization, on the dataset MNIST. The results show that the proposed method still has an accuracy advantage.

Key words: neural network verification; symbolic propagation; abstract interpretation; multi-path back-propagation

神经网络已广泛应用于各种现实场景. 过去难以处理的高维复杂问题, 如图像分类, 神经网络在一定程度上已经可以达到人类的水平. 随着硬件性能提升和算法的进步, 神经网络模型甚至在一些问题上已经十分成熟, 但同时, 其可解释性[1]却十分有限. 另一方面, 由于自身不可避免的过拟合以及攻击技术的发展, 使得无论来自于自然或人为影响, 神经网络都是极其不稳定的. 因此, 在一些安全攸关的应用场景下, 神经网络的

* 基金项目: 深圳市高等院校稳定支持计划(20200810045225001); 国家自然科学基金(62002228); 深圳市科创委基础研究项目(JCYJ20210324094202008)

本文由“智能系统的分析和验证”专题特约编辑明仲教授、张立军教授和秦胜潮教授推荐.

收稿时间: 2021-09-05; 修改时间: 2021-10-14; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

部署应用存在较强的局限性. 比如对于自动驾驶系统, 路标的识别错误可能会导致灾难性的后果. 若将神经网络应用于这样的场景, 其鲁棒性必须有严格的保证.

神经网络鲁棒性保障最直接的方法是使用足够多的测试用例来进行测试, 但有限的测试用例并不能保证其绝对安全性. 如一张 28×28 像素的灰阶图片, 若仅允许对任意像素值最大加 1, 产生的图片已多达 $2^{28\times28}$ 种, 全部测试显然不现实. 面对这样的问题, 神经网络的形式化验证受到了广泛关注, 它能够为神经网络的安全性提供数学保证.

神经网络验证要处理的典型问题是网络的可达性质, 即给定一个神经网络及输入范围, 计算输入范围经过这个网络能够到达的输出范围. 在实际问题中, 比如图像分类问题, 这一性质通常和局部鲁棒性[2]联系在一起. 局部鲁棒性是给定一张具体的输入图片和一个最大扰动范围, 若扰动后的图片分类标签不发生改变, 我们称这个分类网络关于输入图片和扰动范围是鲁棒的; 反之, 若存在分类标签的改变, 则称这个分类网络关于输入图片和扰动范围是不鲁棒的. 这一问题的困难之处在于神经网络中非线性激活函数的叠加, 导致难以准确地求解网络对输入输出的映射关系. 已经有一些综述[3−6]介绍了目前神经网络验证的主流方法和工具, 同时, 能够验证的网络规模也在逐年增加. 然而遗憾的是, 目前依然没有足够有效的方法能够直接应用于工业场景的问题规模. 因此, 发展更加快速和准确的神经网络验证方法备受关注, 也是本文的关注点.

本文着重研究只包含 ReLU 激活函数的神经网络. 目前, 对于 ReLU 神经网络的验证方法大致可分为 2 类.

- 一类方法提供可靠(sound)且完备(complete)的结果, 即我们不仅能够相信它给出的安全答案, 也能相信它给出的不安全答案. 显然, 这要求计算网络在给定输入范围下的准确可达集. 文献[7]证明了这类方法具有 NP 时间复杂度.
- 另一类方法提供可靠却不完备的结果, 即能够相信它给出的安全答案, 然而当这类方法无法给出答案或给出不安全答案时, 我们无法根据这一结果得到网络不安全的结论. 这类方法通常对 ReLU 网络的可达集做上近似处理, 即真正的可达集包含于这类方法计算得到的可达集. 因此, 当这类方法得到的可达集与不安全区域有交集时, 并不能说明真正的可达集与不安全区域有交集. 按照近似方法不同, 这类方法又可大致分为两组: (1) 基于线性抽象, 使用线性抽象将 ReLU 激活函数抽象为包含其凸包的线性约束区域[8,9]; (2) 基于半正定规划(SDP), 将 ReLU 激活函数表示为二次约束, 再将得到的验证问题松弛为 SDP 问题[10,11]. 上近似方法相较于计算准确可达集有较低的时间复杂度. 尤其是线性抽象方法, 它使用容易求解的线性形式近似网络的真正可达集, 能够用于验证较大规模的网络.

与线性抽象紧密相关的另一个概念是符号传播[12]. 符号传播技术在神经网络上传播节点取值的符号约束. 如在图 1 中, 节点 $x_{2,1}$ 的取值依赖 $x_{1,1}$ 和 $x_{1,2}$, 它们的关系是 $x_{2,1}=x_{11}+x_{1,2}$; 而 $x_{3,1}$ 的取值依赖 $x_{2,1}$, 它们的关系是 $x_{3,1}=ReLU(x_{2,1})$. 则代入 $x_{2,1}$ 的符号约束, 可得 $x_{3,1}=ReLU(-x_{1,1}+x_{1,2})$. 符号传播可以用来显式地描述节点之间的取值关系, 是一种有效分析神经网络的方法. 线性抽象则是将上述非线性的关系式 $x_{3,1}=ReLU(-x_{1,1}+x_{1,2})$ 抽象为线性约束, 以降低求解难度.

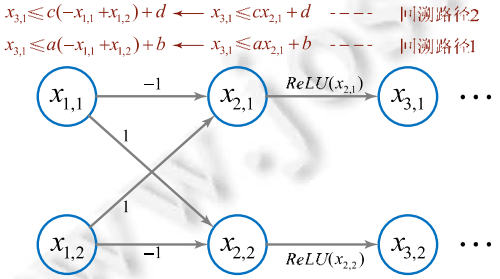


图 1 节点 $x_{3,1}$ 的两条回溯路径

针对基于线性抽象的符号传播方法, 本文提出了多路径回溯的概念. 例如在图 1 中, 如果节点值 $x_{3,1}=ReLU(x_{2,1})$被抽象为区间$[0,ax_{2,1}+b]$, 其中, a, b 为确定常数, 那么 $x_{3,1}$ 的上界可由 $x_{3,1}\leqslant ax_{2,1}+b$ 确定. 利用 $x_{2,1}$ 的

符号约束向输入层替换, 可得到此上界关于输入层的表示 $x_{3,1}\leqslant a(-x_{1,1}+x_{1,2})+b$. 利用符号约束替换到输入层的符号传播过程实际上确定了 $x_{3,1}$ 上界的一条传播路径, 我们称其为回溯路径(back-propagation path), 这类方法的代表是 DeepPoly[13]. 若 $x_{3,1}$ 有另一上界 $x_{3,1}\leqslant cx_{2,1}+d$, 则上述替换过程可确定 $x_{3,1}$ 上界的另一条回溯路径, 进而得到 $x_{3,1}$ 的另一个上界表示. 在多路径回溯的概念下, 已有的方法仅使用单条回溯路径, 因此只能得到 $x_{3,1}$ 的一个上界. 使用多条回溯路径可以得到 $x_{3,1}$ 的多个上界, 从而可能得到比使用单条回溯路径更为精确的上界. 更精确的上下界是这类方法的关键, 它有助于得到更精确的验证结果.

本文对基于线性抽象的符号传播方法作出改进, 旨在提高这类方法的精度. 我们的主要贡献有以下 3 点.

- 我们提出了多路径回溯的概念, 并指出现有的基于线性抽象的符号传播方法仅使用单条回溯路径计算神经网络节点上下界, 是多路径回溯的一种特例.
- 我们实现了多路径回溯方法, 并在数据集 ACAS Xu, MNIST 及 CIFAR10 上与使用单条回溯路径的 DeepPoly 对比. 结果表明, 使用多条回溯路径能明显提升验证精度, 而仅引入较低的额外时间代价.
- 我们在 MNIST 数据集上将多路径回溯方法与使用全局优化方法的 Optimized LiRPA 对比, 结果表明, 多路径回溯方法依然具有精度优势.

本文第 1 节具体介绍一些相关概念和已有方法. 第 2 节提出多路径回溯的概念并证明其可靠性. 第 3 节通过实验评估我们的方法. 第 4 节讨论更多相关工作. 第 5 节给出结论.

1 背景知识

本文关注激活函数为 ReLU 的前馈神经网络的验证问题. 前馈神经网络是节点分层表示的有向无环图, 第 1 层称为输入层, 最后一层称为输出层. 我们用 $x_{i,j}$ 表示第 i 层的第 j 个节点. 为避免引入过多符号, 在不产生歧义的上下文中, 也用 $x_{i,j}$ 表示对应节点的值. 除了 i=1 即输入层外, 每个节点 $x_{i,j}$ 被它的所有上层节点用单向边连接, 这些单向边的权重构成节点 $x_{i,j}$ 的权重向量 $\boldsymbol{w}_{i,j}$. 除权重向量之外, 每个 $x_{i,j}$ 有一个常量偏移 $b_{i,j}$. 我们用 $\boldsymbol{x}_i$ 表示第 i 层节点取值组成的向量. 为了描述方便, 本文将 ReLU 层作为一种层类型加到每个仿射变换层之后, 形成仿射变换层和 ReLU 层交替的神经网络结构, 如图 2 所示. 我们约定: 除输入层和输出层外, 奇数层为 ReLU 层, 偶数层为仿射变换层. 在这种表示下, 一个 ReLU 层节点 $x_{i+1,j}$ 与连接它的仿射变换层节点 $x_{i,j}$ 的关系是 $x_{i+1,j}=ReLU(x_{i,j})\triangleq\max(0,x_{i,j})$; 一个仿射变换层节点 $x_{i,j}$ 与上层节点的关系是 $x_{i,j}=\boldsymbol{w}_{i,j}\boldsymbol{x}_{i-1}+b_{i,j}$. 例如在图 2 中, $x_{3,1}=ReLU(x_{2,1})$, 其中, $x_{2,1}$ 由仿射变换 $x_{2,1}=\boldsymbol{w}_{2,1}\boldsymbol{x}_1+b_{2,1}$ 得到.

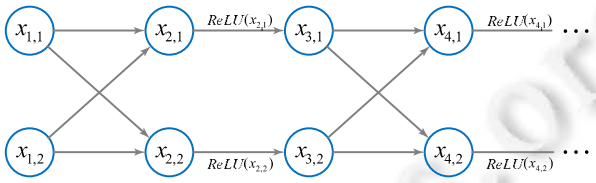


图 2 神经网络结构

给定一个有界范围的输入, 每个节点 $x_{i,j}$ 的取值不再是一个数值而是一个数值区间$[l_{i,j},u_{i,j}]$, 其中, $l_{i,j}$ 和 $u_{i,j}$ 分别称为节点 $x_{i,j}$ 的数值下界和数值上界. 仿射变换层节点 $x_{i,j}$ 作为 ReLU 函数的输入, 若有 $l_{i,j}<0<u_{i,j}$, 则称这个节点激活状态不确定.

一个神经网络确定了一个从输入到输出的函数 f, 从经验上来看, f 通常具有十分复杂的映射关系, 因此, 验证问题最关注的是 f 的可达集性质.

定义 1(验证问题). 给定 L 层神经网络 $f:\mathbb{R}^{d_1}\to\mathbb{R}^{d_L}$, 其中, d_i 表示第 i 层维数; 给定输入范围$\mathcal{X}_1$, 不安全输出范围 $\overline{S}$. 验证问题的目标是, 确定是否有 $f(\mathcal{X}_1)\cap\overline{S}=\varnothing$.

如果可达集 $f(\mathcal{X}_1)$与不安全区域 $\overline{S}$ 交集为空, 称神经网络 f 关于性质 $(\mathcal{X}_1,\overline{S})$ 是安全的; 反之, 若交集非空,

则称关于性质 $(\mathcal{X}_1,\bar{S})$ 是不安全的. 例如图像分类问题中, 输入域$\mathcal{X}_1$ 是对一张原始图片经过一定大小扰动的全体图片, 不安全区域 $\bar{S}$ 可以是除正确分类标签之外的其他标签, 这样, 上述验证问题即图像分类网络的局部鲁棒性问题[2]. 验证 $f(\mathcal{X}_1)$与 $\bar{S}$ 交集为空, 等价于验证在输入范围$\mathcal{X}_1$ 下, 正确标签对应的节点值恒大于其他标签对应的节点值. 因此, 求解验证问题的关键实际上是求解输出层节点的可达集. 对于使用线性抽象等可靠但不完备的方法, 关键则是如何更准确地近似输出层节点的真正可达集.

1.1 线性抽象

线性抽象方法将非线性激活函数 ReLU 抽象为包含其凸包的线性约束区域, 因为处理线性函数的叠加比处理非线性函数的叠加简单. 图 3 表示常见的 4 种 ReLU 抽象方式, 图中的横轴表示一个仿射变换层节点 $x_{i,j}$ 的取值, 它的数值上下界$[l_{i,j},u_{i,j}]$跨过原点, 因此激活状态不确定; 纵轴表示它经过 ReLU 激活函数的取值$x_{i+1,j}$. 蓝色阴影区域表示对 ReLU 函数抽象后的输入输出关系. 如在图 3(a)中, $x_{i,j}$=0 时, $x_{i+1,j}$=*ReLU*($x_{i,j}$)=0 被抽象为蓝色区域与纵轴的交集, 即图 3(a)中黑色方括号区间.

(a) (b) (c) (d)

图 3 4 种 ReLU 抽象方式

图 3(a) 通常称为 LP 抽象[14], 其中, LP 是线性规划(linear programming)的简称. 其抽象区域重合于 ReLU 激活函数在$[l_{i,j},u_{i,j}]$上的二维凸包, 可以表述为以下 3 个线性约束:

$$x_{i+1,j}\leqslant\frac{u_{i,j}}{u_{i,j}-l_{i,j}}(x_{i,j}-l_{i,j}),$$
$$x_{i+1,j}\leqslant x_{i,j},$$
$$x_{i+1,j}\leqslant 0.$$

对单个 ReLU 激活函数的线性抽象中, 这种抽象方法是最精确的. LP 抽象的一个重要问题是: 每个节点具有两个下界约束, 使得确定这个节点的数值上下界实际上成为一个最优化问题, 这也是其名称 LP 抽象的来源. 使用 LP 抽象计算每个节点的数值上下界能够得到较为精确的结果, 但随着网络深度和宽度增加, 求解接近输出层的节点数值上下界对应的 LP 问题会变得十分复杂.

图 3(b)–图 3(d)均为简单线性抽象, 它们分别称为直角三角形抽象、钝角三角形抽象和平行四边形抽象[15]. 在这 3 种抽象方法中, 每个 ReLU 节点的上界和下界约束分别只有一个, 这使得可以利用符号传播逐层地传递上下界约束, 而不涉及复杂的优化方法. 它们是神经网络验证中速度很快的方法, 但另一方面, 较少的约束意味着较大的精度损失. 而且随着网络深度的增加, 精度损失的叠加会愈加明显. 如何利用这种简单的约束得到尽可能高的验证精度, 是目前备受关注的问题.

下面的定理 1 说明这 4 种抽象的可靠性, 即 $x_{i,j}$ 经过 ReLU 的值属于经过上述 ReLU 抽象的区间.

定理 1. 给定 $l_{i,j}$<0, $u_{i,j}$>0, 对于任意$\lambda\in[0,1]$及任意 $x_{i,j}\in[l_{i,j},u_{i,j}]$, 下式成立:

$$\lambda x_{i,j}\leqslant ReLU(x_{i,j})\leqslant\frac{u_{i,j}}{u_{i,j}-l_{i,j}}(x_{i,j}-l_{i,j}).$$

证明: 注意到 $u_{i,j}/(u_{i,j}-l_{i,j})\in(0,1)$且 $x_{i,j}-l_{i,j}\geqslant 0$. 当 $x_{i,j}$<0 时, 由 ReLU 函数定义, *ReLU*($x_{i,j}$)=0, 而不等式左边 $\lambda x_{i,j}\leqslant 0$ 恒成立且不等式右边 $u_{i,j}(x_{i,j}-l_{i,j})/(u_{i,j}-l_{i,j})\geqslant 0$, 故上式成立; 当 $x_{i,j}\geqslant 0$ 时, *ReLU*($x_{i,j}$)=$x_{i,j}$, 而不等式左边 $\lambda x_{i,j}\leqslant x_{i,j}$恒成立且 $u_{i,j}(x_{i,j}-l_{i,j})/(u_{i,j}-l_{i,j})-ReLU(x_{i,j})=l_{i,j}(x_{i,j}-u_{i,j})/(u_{i,j}-l_{i,j})\geqslant 0$, 故上式成立. □

取λ=0 及λ=1 时, 定理 1 蕴含 LP 抽象的可靠性; 取$\lambda=u_{i,j}/(u_{i,j}-l_{i,j})$时, 蕴含平行四边形抽象的可靠性. 同理可得直角三角形和钝角三角形抽象的可靠性. 此外, 定理 1 还表明, 任意$\lambda x_{i,j}$都可以作为 *ReLU*($x_{i,j}$)的下界, 从

而与上界 $u_{i,j}(x_{i,j}-l_{i,j})/(u_{i,j}-l_{i,j})$构成仅有一个线性下界约束和一个线性上界约束的 ReLU 可靠抽象.

1.2 符号传播和DeepPoly

文献[12]引入了符号传播方法在神经网络验证问题中的应用. 目前, 主流的验证方法都会使用符号传播方法维护每个节点与输入层节点的约束关系. 这可以通过前向传递符号约束得到, 也可以反向替换符号约束至输入层得到. 我们将反向替换符号约束至输入层的方法称为回溯(back-propagation). DeepPoly[13]是使用回溯方法的代表, 它使用简单线性抽象处理 ReLU 函数. 对于每个节点 $x_{i,j}$, 它维护 $x_{i,j}$ 关于上层节点值 $\boldsymbol{x}_{i-1}$ 的符号约束, 然后通过代入 $\boldsymbol{x}_{i-1}$ 关于 $\boldsymbol{x}_{i-2}$ 的符号约束, 依次向输入层回溯, 得到 $x_{i,j}$ 关于输入层 $\boldsymbol{x}_1$ 表示的上下界符号约束. 最后, 通过代入 $\boldsymbol{x}_1$ 的数值上下界, 计算 $x_{i,j}$ 的数值上下界. 回溯到输入层可以得到比前向传递符号约束更精确的上下界.

图 4 展示了 DeepPoly 计算每个节点数值上下界的过程. 网络输入范围$\mathcal{X}=\{(x_{1,1},x_{1,2}):-1\leqslant x_{1,1}\leqslant 1\wedge -1\leqslant x_{1,2}\leqslant 1\}$且各节点偏移都为 0, 节点权重在前向边上标出. 注意, 节点 $x_{4,1}$ 由 $x_{3,1}$ 和 $x_{3,2}$ 经过仿射变换得到,它们之间的关系是 $x_{4,1}\geqslant x_{3,1}-x_{3,2}$ 及 $x_{4,1}\leqslant x_{3,1}-x_{3,2}$, 这是 $x_{4,1}$ 取值的符号约束. 为了得到 $x_{4,1}$ 尽可能精确的数值上下界, 需要得到它关于输入层的符号表示, 因此利用 $x_{3,1}$ 和 $x_{3,2}$ 向输入层回溯. $x_{3,1}$ 由 $x_{2,1}$ 经过 ReLU 函数得到,假设这里使用直角三角形抽象, 则有 $x_{3,1}\geqslant 0$ 以及 $x_{3,1}\leqslant 0.5x_{2,1}+1$; 同理, 对于 $x_{3,2}$ 有 $x_{3,2}\geqslant 0$ 以及 $x_{3,2}\leqslant 0.5x_{2,2}+1$. 那么 $x_{4,1}$ 的上界是 $x_{4,1}\leqslant 0.5x_{2,1}+1-0$, 下界是 $x_{4,1}\geqslant 0-(0.5x_{2,2}+1)$. 按照同样的方法继续向输入层回溯, 可以得到 $x_{4,1}$ 关于输入层表示的符号约束为 $x_{4,1}\geqslant -0.5x_{1,1}+0.5x_{1,2}-1$ 及 $x_{4,1}\leqslant -0.5x_{1,1}+0.5x_{1,2}+1$, 最后代入 $x_{1,1}$ 和 $x_{1,2}$ 的数值上下界得到 $x_{4,1}\geqslant -2$ 及 $x_{4,1}\leqslant 2$. 因此, [−2,2]作为 $x_{4,1}$ 的数值上下界.

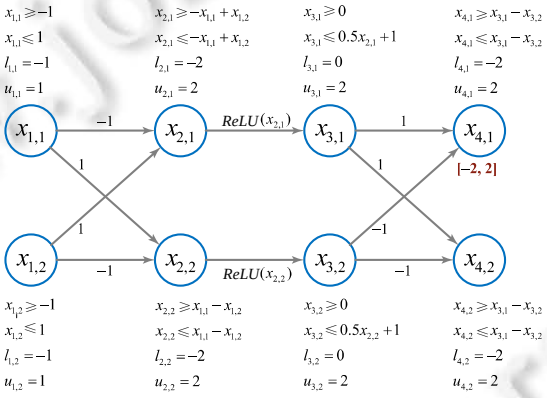


图 4 使用 DeepPoly 计算节点上下界

对于所有非输入层节点, DeepPoly 都使用这种回溯到输入层的方式计算其数值上下界. 除了回溯到输入层, DeepPoly 还使用一个有效的启发式策略来减少对每个 ReLU 激活函数的抽象误差: 对于激活状态不确定的节点, 例如 $x_{2,1}$, 如果求得其数值上下界为$[l_{2,1},u_{2,1}]$, 若$|l_{2,1}|\geqslant u_{2,1}$, 则选择直角三角形抽象, 因为此时直角三角形的面积为 $0.5\times u_{2,1}(u_{2,1}-l_{2,1})$, 小于等于钝角三角形的面积 $0.5\times|l_{2,1}|(u_{2,1}-l_{2,1})$; 否则, 若$|l_{2,1}|<u_{2,1}$, 则选择钝角三角形抽象, 因为此时钝角三角形的面积小于直角三角形. 更小的面积意味着更小的抽象误差.

DeepPoly 得到每个节点的数值上下界是可靠的, 即对于每个节点, 它真正的数值上下界包含于 DeepPoly 计算得到的上下界. 因此, 通过这种方法可以快速得到真正可达集的一个上近似. 如果一个输入范围被 DeepPoly 验证是安全的, 那么蕴含它确实是安全的; 但反之不一定成立. 上近似的精度决定了这种方法的验证能力.

下一节将介绍我们的方法, 它使用多条回溯路径, 能够改善对真正可达集上近似的精度, 而仅引入较低的额外时间开销.

2 多路径回溯

从第 1.2 节可以看出, DeepPoly 对每个节点 $x_{i,j}$ 维护一组符号上下界约束. 一组符号约束最多构成一条回溯路径, 因此它仅能得到 $x_{i,j}$ 的一个数值上下界. 如果对每个节点 $x_{i,j}$ 维护多组符号上下界约束, 分别在多组符号约束构成的多条回溯路径上回溯, 则可得到 $x_{i,j}$ 的多个数值上下界. 不同回溯路径上可以使用不同特点的线性抽象产生不同的数值上下界, 定理 1 保证这多个数值上下界的交集也是 $x_{i,j}$ 的数值上下界, 因此我们可以得到更精确的上下界, 从而进一步构造更精确的上近似. 这是本文的主要想法.

给定正整数 n, 定义$[n]=\{1,2,\ldots,n\}$表示前 n 个正整数构成的集合. 用$\mathbb{R}_1[\boldsymbol{x}]$表示实系数线性多项式集合. 给定 $m\in[M]$, 我们用抽象元素 $a_{i,j,m}=(a^{\geqslant}_{i,j,m},a^{\leqslant}_{i,j,m})\in\mathcal{A}=\mathbb{R}_1[\boldsymbol{x}]\times\mathbb{R}_1[\boldsymbol{x}]$ 表示节点 $x_{i,j}$ 的符号上下界约束, 且限定 $a^{\geqslant}_{i,j,m}$ 和 $a^{\leqslant}_{i,j,m}$ 具有以下特定形式:

$$a^{\geqslant}_{i+1,j,m},a^{\leqslant}_{i+1,j,m}=\sum_{k=1}^{d_i}q_{i+1,k,m}x_{i,k}+r_{i+1,j,m},$$

$$a^{\geqslant}_{1,j,m}=a^{\leqslant}_{1,j,m}=x_{1,j}.$$

上式中, $q_{i+1,k,m}$ 和 $x_{i+1,j,m}$ 为常数, $a^{\geqslant}_{i+1,j,m}$ 和 $a^{\leqslant}_{i+1,j,m}$ 分别是 $x_{i+1,j}$ 关于第 i 层节点表示的线性下界约束和线性上界约束. 通过 $x_{i+1,j}$ 所依赖的所有节点的符号约束, 可以回溯到输入层得到 $x_{i+1,j}$ 关于输入层的线性表示. 定义变换函数$\gamma^{\geqslant},\gamma^{\leqslant}:\mathbb{R}_1[\boldsymbol{x}]\to\mathbb{R}_1[\boldsymbol{x}]$如下:

$$\begin{aligned}\gamma^{\geqslant}\left(\sum_{k=1}^{d_i}q_{i+1,k,m}x_{i,k}+r_{i+1,j,m}\right)&=\sum_{k=1}^{d_i}\gamma^{\geqslant}(q_{i+1,k,m}x_{i,k})+r_{i+1,j,m}\\&=\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})+1|}{2}q_{i+1,k,m}\gamma^{\geqslant}(a^{\geqslant}_{i,k,m})+\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})-1|}{2}q_{i+1,k,m}\gamma^{\leqslant}(a^{\leqslant}_{i,k,m})+r_{i+1,j,m},\\\gamma^{\leqslant}\left(\sum_{k=1}^{d_i}q_{i+1,k,m}x_{i,k}+r_{i+1,j,m}\right)&=\sum_{k=1}^{d_i}\gamma^{\leqslant}(q_{i+1,k,m}x_{i,k})+r_{i+1,j,m}\\&=\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})+1|}{2}q_{i+1,k,m}\gamma^{\leqslant}(a^{\leqslant}_{i,k,m})+\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})-1|}{2}q_{i+1,k,m}\gamma^{\geqslant}(a^{\geqslant}_{i,k,m})+r_{i+1,j,m},\end{aligned}$$

$\gamma^{\geqslant}(x_{1,k})=\gamma^{\leqslant}(x_{1,k})=x_{1,k}$, 对任意$k$.

上式中, δ为符号函数, 当 $q\geqslant 0$ 时, $\delta(q)=+1$; 当 $q<0$ 时, $\delta(q)=-1$. $\gamma^{\geqslant}$和$\gamma^{\leqslant}$是一组互递归函数, 它们用于描述 $x_{i+1,j}$ 回溯到输入层的过程, 最终得到关于输入层表示的符号下界和符号上界. 在求 $x_{i+1,j}$ 的下界时, 如果 $x_{i,k}$ 对应的系数 $q_{i+1,k,m}$ 为正, 需要使用 $x_{i,k}$ 的符号下界; 如果 $x_{i,k}$ 对应的系数 $q_{i+1,k,m}$ 为负, 则需要使用 $x_{i,k}$ 的符号上界. 求 $x_{i+1,j}$ 的上界时同理.

整个递归过程实际上确定了一条回溯路径. 定义函数 $\gamma(a_{i,j,m})=(\gamma^{\geqslant}(a^{\geqslant}_{i,j,m}),\gamma^{\leqslant}(a^{\leqslant}_{i,j,m})):\mathcal{A}\to\mathcal{A}$, 对任意 $m\in[M]$, $\gamma(a_{i,j,m})$确定了使用 $x_{i,j}$ 的第 m 组符号约束计算数值上下界的回溯路径:

$$a_{1,*,m}\leftarrow a_{2,*,m}\leftarrow\ldots\leftarrow a_{i-1,*,m}\leftarrow a_{i,j,m}.$$

由 M 组抽象元素 $a_{i,j,m}$ 可以确定 M 条回溯路径, 通过这 M 条回溯路径, 可得到 $x_{i,j}$ 的 M 个关于输入层表示的符号上下界$\gamma(a_{i,j,m})$. 定义从$\gamma(a_{i,j,m})$得到数值上下界的具象函数 $\gamma^{\star}:\mathcal{A}\to\mathbb{R}\times\mathbb{R}$ 如下:

$$\gamma^{\star}(\gamma(a_{i,j,m}))=\gamma^{\star}(\gamma^{\geqslant}(a^{\geqslant}_{i,j,m}),\gamma^{\leqslant}(a^{\leqslant}_{i,j,m}))=(\gamma^{\star}_{\geqslant}(\gamma^{\geqslant}(a^{\geqslant}_{i,j,m})),\gamma^{\star}_{\leqslant}(\gamma^{\leqslant}(a^{\leqslant}_{i,j,m}))),$$

其中,

$$\begin{aligned}\gamma^{\star}_{\geqslant}(\gamma^{\geqslant}(a^{\geqslant}_{i,j,m}))&=\gamma^{\star}_{\geqslant}\left(\sum_{k=1}^{d_1}q_{1,k,m}x_{1,k}+r_{1,m}\right)=\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})+1|}{2}q_{1,k,m}l_{1,k}+\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})-1|}{2}q_{1,k,m}u_{1,k}+r_{1,m},\\\gamma^{\star}_{\leqslant}(\gamma^{\leqslant}(a^{\leqslant}_{i,j,m}))&=\gamma^{\star}_{\leqslant}\left(\sum_{k=1}^{d_1}q_{1,k,m}x_{1,k}+r_{1,m}\right)=\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})+1|}{2}q_{1,k,m}u_{1,k}+\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})-1|}{2}q_{1,k,m}l_{1,k}+r_{1,m}.\end{aligned}$$

对于 $x_{i,j}$ 的数值下界, 若 $q_{1,k,m}>0$, 使用 $x_{1,k}$ 的下界; 若 $q_{1,k,m}<0$, 则使用 $x_{1,k}$ 的上界. $x_{i,j}$ 数值上界的计算类似.

为了简便，我们记 $\gamma^*(\gamma(a_{i,j,m}))$ 为 $\gamma^*(a_{i,j,m})$. 则 $x_{i,j}$ 可计算出 M 个数值上下界 $(l_{i,j,m},u_{i,j,m})=\gamma^*(a_{i,j,m})$. 它们的交集也是 $x_{i,j}$ 的数值上下界，我们取:

$$[l_{i,j},u_{i,j}]=\bigcap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$$

作为节点 $x_{i,j}$ 最终的数值上下界.

抽象元素 $a_{i,j,m}$ 的计算取决于神经网络中 $x_{i,j}$ 的节点类型. 对于第 $m\in[M]$种抽象方式，若 $x_{i,j}$ 是激活状态不确定的节点，我们定义其抽象元素 $a_{i,j,m}$ 的抽象变换为

$$ReLU_m^{\#}(a_{i,j,m})=ReLU_m^{\#}((a_{i,j,m}^{\geqslant},a_{i,j,m}^{\leqslant}))=(a_{i+1,j,m}^{\geqslant},a_{i+1,j,m}^{\leqslant}),$$

其中，$(a_{i+1,j,m}^{\geqslant},a_{i+1,j,m}^{\leqslant})$ 表示节点 $x_{i+1,j}$ 对应的抽象元素，具体形式为

$$a_{i+1,j,m}^{\geqslant}=\lambda_m x_{i,j},\ a_{i+1,j,m}^{\leqslant}=\frac{u_{i,j}}{u_{i,j}-l_{i,j}}(x_{i,j}-l_{i,j}).$$

上式中，$\lambda_m\in[0,1]$为常数. 即在每条回溯路径上，我们对激活状态不确定的节点使用相同的λ_m. 对于激活状态确定的 ReLU 以及仿射变换函数，容易定义其准确的抽象变换，本文省略其具体形式.

DeepPoly 对每个节点 $x_{i,j}$ 仅维护一组抽象元素 $a_{i,j,1}$，是 M=1 的特例，其λ_1 对应第 1.2 节所述的启发式策略，我们称它确定的回溯路径为 DeepPoly 路径. 使用一条回溯路径仅能得到 $x_{i,j}$ 的一个数值上下界$[l_{i,j,1},u_{i,j,1}]$. 通过维护 $x_{i,j}$ 的多组抽象元素，利用这多组抽象元素构成的多条回溯路径，我们可以得到 $x_{i,j}$ 更多的数值上下界信息. 此外，多条回溯路径蕴含至少有任何一条回溯路径的效果，则至少达到 DeepPoly 的精确程度.

图 5 中网络是图 4 的扩展，它展示了使用两条回溯路径计算每个节点数值上下界的过程.

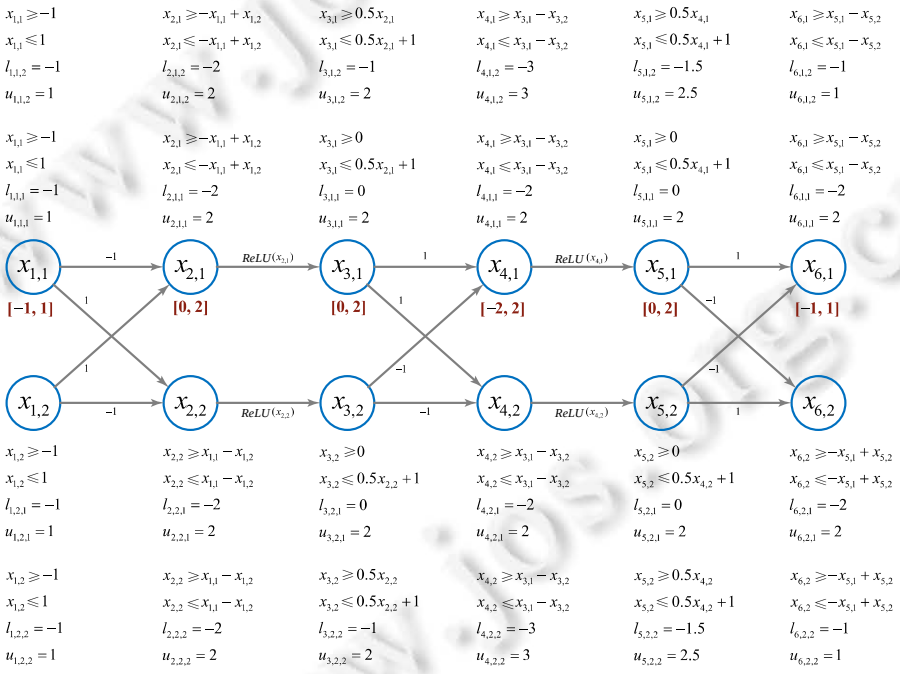


图 5　使用两条回溯路径计算节点上下界

其中，m=1 对应 DeepPoly 路径，见网络外侧第 1 行–第 4 行；m=2 对应平行四边形抽象的回溯路径，见网络外侧第 5 行–第 8 行. 网络的输入范围为$\mathcal{X}_1=\{(x_{1,1}+x_{1,2}):-1\leqslant x_{1,1}\leqslant 1\wedge -1\leqslant x_{1,2}\leqslant 1\}$且每个节点的偏移为 0. 对于 ReLU 层节点 $x_{3,1}$，我们维护经过 DeepPoly 抽象得到的符号约束 $x_{3,1}\geqslant 0$ 及 $x_{3,1}\leqslant 0.5x_{2,1}+1$；同时维护经过平行四边形抽象得到的符号约束 $x_{3,1}\geqslant 0.5x_{2,1}$ 及 $x_{3,1}\leqslant 0.5x_{2,1}+1$. 这两组符号约束作为 $x_{3,1}$ 的两个抽象元素构成两

条不同回溯路径. 对于 DeepPoly 路径, 其关于输入层的符号上下界为

$$\begin{aligned}\gamma(a_{3,1,1}) &= (\gamma^{\geqslant}(a^{\geqslant}_{3,1,1}), \gamma^{\leqslant}(a^{\leqslant}_{3,1,1}))\\ &= (\gamma^{\geqslant}(0), \gamma^{\leqslant}(0.5x_{2,1}+1))\\ &= (0, 0.5\gamma^{\leqslant}(a^{\leqslant}_{2,1,1})+1)\\ &= (0, 0.5\gamma^{\leqslant}(-x_{1,1}+x_{1,2})+1)\\ &= (0, 0.5(-\gamma^{\geqslant}(a^{\geqslant}_{1,1,1})+\gamma^{\leqslant}(a^{\leqslant}_{1,2,1}))+1)\\ &= (0, 0.5(-x_{1,1}+x_{1,2})+1).\end{aligned}$$

对应的数值下界和上界分别为

$$l_{3,1,1} = \gamma^{\star}_{\geqslant}(0) = 0,$$

$$u_{3,1,1} = \gamma^{\star}_{\leqslant}(0.5(-x_{1,1}+x_{1,2})+1) = -0.5l_{1,1}+0.5u_{1,2}+1 = 2.$$

对于平行四边形抽象对应的回溯路径, 其关于输入层的符号上下界为

$$\begin{aligned}\gamma(a_{3,1,2}) &= (\gamma^{\geqslant}(a^{\geqslant}_{3,1,2}), \gamma^{\leqslant}(a^{\leqslant}_{3,1,2}))\\ &= (\gamma^{\geqslant}(0.5x_{2,1}), \gamma^{\leqslant}(0.5x_{2,1}+1))\\ &= (0.5\gamma^{\geqslant}(-x_{1,1}+x_{1,2}), 0.5(-x_{1,1}+x_{1,2})+1)\\ &= (0.5(-\gamma^{\leqslant}(a^{\leqslant}_{1,1,2})+\gamma^{\geqslant}(a^{\geqslant}_{1,2,2})), 0.5(-x_{1,1}+x_{1,2})+1)\\ &= (0.5(-x_{1,1}+x_{1,2}), 0.5(-x_{1,1}+x_{1,2})+1).\end{aligned}$$

对应的数值下界和上界分别为

$$l_{3,1,2} = \gamma^{\star}_{\geqslant}(0.5(-x_{1,1}+x_{1,2})) = -0.5u_{1,1}+0.5l_{1,2} = -1,$$

$$u_{3,1,2} = \gamma^{\star}_{\leqslant}(0.5(-x_{1,1}+x_{1,2})+1) = -0.5l_{1,1}+0.5u_{1,2}+1 = 2.$$

我们取 $x_{3,1}$ 最终的数值上下界为

$$[l_{3,1},u_{3,1}]=[l_{3,1,1},u_{3,1,1}]\cap[l_{3,1,2},u_{3,1,2}]=[0,2].$$

按照同样的做法, 可得到 $x_{4,1}$ 的数值上下界为$[l_{4,1},u_{4,1}]=[-2,2]$. 因此, $x_{4,1}$ 是激活状态不确定的节点, 其对应的抽象变换为

$$ReLU^{\#}_1(a_{4,1,1}) = a_{5,1,1} = (0, 0.5x_{4,1}+1),$$

$$ReLU^{\#}_2(a_{4,1,2}) = a_{5,1,2} = (0.5x_{4,1}, 0.5x_{4,1}+1).$$

由 $a_{5,1,1}$ 和 $a_{5,1,2}$ 可按照上述方式得到 $x_{5,1}$ 的数值上下界为$[0,2]$. 在节点 $x_{6,1}$ 处, 使用两条回溯路径已经能够得到比 DeepPoly 更为精确的结果, 因为有:

$$l_{6,1,1} = \gamma^{\star}_{\geqslant}(0.25x_{1,1}-0.25x_{1,2}-1.5) = -2,$$

$$u_{6,1,1} = \gamma^{\star}_{\leqslant}(-0.25x_{1,1}+0.25x_{1,2}+1.5) = 2,$$

$$l_{6,1,2} = \gamma^{\star}_{\geqslant}(-1) = -1,$$

$$l_{6,1,2} = \gamma^{\star}_{\geqslant}(-1) = -1.$$

我们取$[l_{6,1},u_{6,1}]=[-2,2]\cap[-1,1]=[-1,1]$, 它包含于 DeepPoly 得到的数值上下界. 更精确的数值上下界是我们的首要目标, 因为它还可以用来构造更精确的抽象, 从而更精确地近似真正的可达集.

下面证明上述方法的可靠性, 即任意节点 $x_{i,j}$ 的任意回溯路径所对应的数值上下界 $\gamma^{\star}(a_{i,j,m})$, 都是此节点在 ReLU 网络中数值上下界的上近似, 这保证我们得到的可达集是 ReLU 网络的上近似.

注意到 $a_{i,j,m}$ 来自对 ReLU 以及仿射变换函数的抽象变换, 对于激活状态确定的 ReLU 以及仿射变换函数, 其抽象变换是准确的. 这里只对抽象变换 $ReLU^{\#}_m$ 证明可靠性, 其他抽象变换同理. 假设激活状态确定的 ReLU 以及仿射变换抽象已经具有可靠性.

定理 2. 设 i 为小于网络层数 L 的任意偶数, 对于任意 $j\in[d_i]$和任意 $m\in[M]$, 上述方法得到的数值上下界:

$$[l_{i+1,j,m},u_{i+1,j,m}]=[\gamma^{\star}_{\geqslant}(\gamma^{\geqslant}(ReLU^{\#}_{m}(a^{\geqslant}_{i,j,m}))),\gamma^{\star}_{\leqslant}(\gamma^{\leqslant}(ReLU^{\#}_{m}(a^{\leqslant}_{i,j,m})))].$$

关于 ReLU 网络中节点 $x_{i+1,j}$ 的数值上下界可靠.

证明: 归纳法. 初始情况 i=2 时, 显然成立, 因为仿射变换的抽象是可靠的. 假设 i=k–2 时上式成立, 其中, k 为大于等于 4 的偶数. 即 $x_{k-1,j}$ 的数值上下界 $[l_{k-1,j},u_{k-1,j}]=\bigcap_{m=1}^{M}[l_{k-1,j,m},u_{k-1,j,m}]$ 是可靠的. 则由仿射变换抽象的可靠性可知, $[l_{k,j},u_{k,j}]=\bigcap_{m=1}^{M}[l_{k,j,m},u_{,j,m}]$ 关于 ReLU 网络中节点 x_{kj} 的数值上下界可靠. 因为

$$\begin{aligned}\gamma(ReLU^{\#}_{m}(a_{k,j,m}))&=(\gamma^{\geqslant}(a^{\geqslant}_{k+1,j,m}),\gamma^{\leqslant}(a^{\leqslant}_{k+1,j,m}))\\&=\left(\gamma^{\geqslant}(\lambda_m x_{k,j}),\gamma^{\leqslant}\left(\frac{u_{k,j}}{u_{k,j}-l_{k,j}}(x_{k,j}-l_{k,j})\right)\right)\\&=\left(\lambda_m\gamma^{\geqslant}(a^{\geqslant}_{k,j,m}),\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\leqslant}(a^{\leqslant}_{k,j,m})-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right),\end{aligned}$$

所以有:

$$\begin{aligned}[l_{i+1,j,m},u_{i+1,j,m}]&=\left[\gamma^{\star}_{\geqslant}(\lambda_m\gamma^{\geqslant}(a^{\geqslant}_{k,j,m})),\gamma^{\star}_{\leqslant}\left(\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\leqslant}(a^{\leqslant}_{k,j,m})-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right)\right]\\&=\left[\lambda_m\gamma^{\star}_{\geqslant}(\gamma^{\geqslant}(a^{\geqslant}_{k,j,m})),\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\star}_{\leqslant}(\gamma^{\leqslant}(a^{\leqslant}_{k,j,m}))-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right]\\&=\left[\lambda_m l_{k,j,m},\frac{u_{k,j}}{u_{k,j}-l_{k,j}}u_{k,j,m}-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right]\\&\supseteq\left[\lambda_m l_{k,j},\frac{u_{k,j}}{u_{k,j}-l_{k,j}}u_{k,j}-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right].\end{aligned}$$

由定理 1 可知, 若$[l_{kj},u_{kj}]$关于 ReLU 网络中 x_{kj} 可靠, 则最后一行关于 ReLU 网络中 $x_{i+1,j}$ 可靠, 即定理 2 成立. □

直观而言, 定理 2 说明: 对于任意激活状态不确定的节点 $x_{i,j}$, 通过其 M 组符号约束回溯到输入层得到的数值上下界关于 ReLU 网络都是可靠的. 于是, 我们的方法得到每个节点真正可达集的上近似.

算法 1 给出了上述方法的伪代码, 算法输入为网络 f、输入范围$\mathcal{X}_1$ 以及对应的不安全区域 $\overline{S}$, 算法判断网络 f 关于 $(\mathcal{X}_1,\overline{S})$ 是否安全. 因为我们把仿射变换和 ReLU 变换作为分离的两层, 所以需要根据不同的层类型来表示 $x_{i,j}$. 在第 3 行, 如果第 i 层为仿射变换层, 那么需要用 i–1 层节点仿射变换表示 $x_{i,j}$. 这里, $\boldsymbol{w}_{i,j}$ 和 $b_{i,j}$ 分别表示节点 $x_{i,j}$ 的权重向量和偏移, $\boldsymbol{x}_{i-1}$ 表示 i–1 层所有节点构成的向量. 仿射变换的抽象表示是准确的, 我们把这个准确的约束作为其抽象 $a_{i,j,m}$. 利用每个 $a_{i,j,m}$ 回溯可得到 M 个数值上下界$[l_{i,j,m},u_{i,j,m}]$, 取它们的交集作为 $x_{i,j}$ 最终的数值上下界, 如第 4 行–第 7 行所示. 当第 i 层节点的数值上下界计算完成, 就可以得到本层节点的可达集$\mathcal{X}_i$, 如第 13 行所示. 对于仿射变换层节点, 其更精确的可达集在第 10 行被用来构造更精确的 M 种 ReLU 抽象. 按照相同的方法, 在第 12 行得到 ReLU 层节点的数值上下界. 第 14 行得到输出层节点可达集后, 就可以判断网络 f 可达集与不安全区域 $\overline{S}$ 是否有交集: 如果没有交集, 返回 SAFE, 说明 f 关于 $(\mathcal{X}_1,\overline{S})$ 安全,; 否则, 返回 UNKNOWN, 因为此时不能保证 f 真正的可达集与 $\overline{S}$ 有交集.

算法 1. 验证一个网络关于一个输入范围是否安全.

输入: 网络 f, 输入范围$\mathcal{X}_1$ 和不安全区域 $\overline{S}$.

输出: 如果安全, 返回 SAFE; 否则, 返回 UNKNOWN.

1. **for** i←1 to L **do** //L 为网络 f 的层数
2. **for** j←1 to d_i **do** //d_i 为第 i 层节点数
3. **if** *layer_type*[i]=*AFFINE* **then** //如果第 i 层为仿射变换层

4. **for** $m\leftarrow 1$ to M **do** //构造仿射变换函数抽象
5. $a_{i,j,m}\leftarrow(\boldsymbol{w}_{i,j}\boldsymbol{x}_{i-1}+b_{i,j},\boldsymbol{w}_{i,j}\boldsymbol{x}_{i-1}+b_{i,j})$
6. $[l_{i,j,m},u_{i,j,m}]\leftarrow[\gamma^{\star}_{\geqslant}(\gamma^{\geqslant}(a^{\geqslant}_{i,j,m})),\gamma^{\star}_{\leqslant}(\gamma^{\leqslant}(a^{\leqslant}_{i,j,m}))]$
7. $[l_{i,j},u_{i,j}]\leftarrow\cap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$ //取 M 个数值上下界的交集
8. **if** *layer_type*[i]=*RELU* **then** //如果第 i 层为 ReLU 层
9. **for** $m\leftarrow 1$ to M **do** //对于 $x_{i,j}$,用 $x_{i\text{-}1,j}$ 构造 M 种 ReLU 抽象
10. $a_{i,j,m}\leftarrow ReLU^{\#}_{m}(a_{i-1,j,m})$
11. $[l_{i,j,m},u_{i,j,m}]=[\gamma^{\star}_{\geqslant}(\gamma^{\geqslant}(a^{\geqslant}_{i,j,m})),\gamma^{\star}_{\leqslant}(\gamma^{\leqslant}(a^{\leqslant}_{i,j,m}))]$
12. $[l_{i,j},u_{i,j}]\leftarrow\cap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$
13. $\mathcal{X}_i\leftarrow[\boldsymbol{L}_i,\boldsymbol{u}_i]$ //得到$\mathcal{X}_i$为第 i 层节点的可达集
14. **if** $\mathcal{X}_L\cap\overline{S}=\varnothing$ **then** //如果输出层可达集与不安全区域交集为空
15. **return** SAFE
16. **else**
17. **return** UNKNOWN

上近似方法(如算法 1)返回 UNKNOWN 时, 说明它得到的可达集$\mathcal{X}_L$与$\overline{S}$有交集, 但$\mathcal{X}_L$是放大的可达集. 虽然无法确定真正的可达集是否与$\overline{S}$存在交集, 然而可以利用求解过程中得到的信息进一步求得更为精确的可达集, 这一过程称为精化. 文献[16,17]均采用这种想法, 以得到更为准确的验证结果.

算法 1 继承了基于简单线性抽象的符号传播方法的高效性, 其时间复杂度为$\mathcal{O}(MN^2)$, 其中, N 为网络 f 的节点数量. 对于每个节点, 回溯计算其数值上下界的时间代价是$\mathcal{O}(N)$. 回溯路径的数量 M 一般取低值常数, 我们将在第 3.2 节讨论关于 M 的取值. 算法 1 的空间复杂度是$\mathcal{O}(N)$, 对每个节点保存常数项个约束.

理论上, 回溯路径的数量可以任意多, 即 M 值可以任意大. 那么一个值得关心的问题是, 任意多的回溯路径能够达到多好的精度? 在第 3.2 节, 我们研究了随 M 值增加, 精度的改善效果.

3 实验与评估

和绝大部分工作一样, 我们关注对具体输入的无穷范数扰动, 并关注给定网络在无穷范数扰动下的鲁棒性. 首先, 无穷范数扰动由无穷范数距离定义.

定义 2(无穷范数距离). 给定两个 n 维向量 $\boldsymbol{x}=(a_i)$, $\boldsymbol{z}=(b_i)$, 其中, $1\leqslant i\leqslant n$, 它们的无穷范数距离 $d_{\infty}(\boldsymbol{x},\boldsymbol{z})$被定义为 $d_{\infty}(\boldsymbol{x},\boldsymbol{z})=\max\limits_{1\leqslant i\leqslant n}|a_i-b_i|$.

通常, 输入范围$\mathcal{X}_1$由上述度量给出, 即给定具体输入 $\boldsymbol{x}$ 和扰动大小θ, $\mathcal{X}_1=\mathcal{B}_{\infty}(\boldsymbol{x},\theta)\triangleq\{\boldsymbol{z}|d_{\infty}(\boldsymbol{x},\boldsymbol{z})\leqslant\theta\}$. 直观而言, $\mathcal{X}_1$ 是 $\boldsymbol{x}$ 每个维度最大允许加或减θ所构成的向量集.

对于分类网络的验证问题, 若给定输入, 鲁棒半径能够很好地量化比较不同上近似方法的精度. 下面是无穷范数距离下鲁棒半径的定义, 在后文中简称为鲁棒半径.

定义 3(d_{∞}鲁棒半径). 对于分类网络 f, 给定输入 $\boldsymbol{x}$ 及对应标签 $label(\boldsymbol{x})$, f 关于 $\boldsymbol{x}$ 的 d_{∞}鲁棒半径被定义为

$$R_f(\boldsymbol{x})=\max\{\{\theta\geqslant 0:f(\boldsymbol{z})=label(\boldsymbol{x}),\forall\boldsymbol{z}\in\mathcal{B}_{\infty}(\boldsymbol{x},\theta)\}\cup\{0\}\}.$$

对于分类网络, 上述定义中, $f(\boldsymbol{z})=label(\boldsymbol{x})$蕴含我们对不安全区域$\overline{S}$的选择是 $\boldsymbol{x}$ 正确标签以外的所有标签, 即针对无目标攻击[18]的鲁棒性. 直观而言, 网络 f 关于输入 $\boldsymbol{x}$ 的鲁棒半径, 就是在不产生对抗样本的前提下, 能对 $\boldsymbol{x}$ 施加的最大扰动θ.

若 f 是归一化输入的网络, 则鲁棒半径$R_f(\boldsymbol{x})\in[0,1]$. 通常使用二分法计算$R_f(\boldsymbol{x})$的值: 取$\theta_1=1/2$作为初始值, 在第 i 次迭代中, 确定网络 f 关于扰动θ_i后的输入范围是否鲁棒.

- 如果不鲁棒, 将$\theta_{i+1}=\theta_i-(1/2)^{i+1}$作为下一个取值;

- 如果鲁棒, 则将$\theta_{i+1}=\theta_i+(1/2)^{i+1}$作为下一个取值.

重复上述迭代过程, 直至找到满足精度要求的θ_i, 将其作为鲁棒半径$R_f(\boldsymbol{x})$的值.

给定神经网络和输入, 上近似方法计算得到的鲁棒半径一般小于真正的鲁棒半径, 因为上近似产生的误差通常会放大求得的可达集. 因此, 可以用不同方法计算得到的鲁棒半径来定量比较这些方法的精确度. 如果某种上近似方法得到的鲁棒半径更大, 即更接近真正的鲁棒半径, 则说明该上近似方法更精确.

我们将提出的多路径回溯方法实现为 AbstraCMP 工具, 其命名来源于该方法在神经网络各节点处对几种抽象(abstraction)的回溯结果进行比较(CMP). AbstraCMP 使用 Python 3.9 实现, 其源代码以及实验结果可以在文献[19]获取. 本节我们在数据集 ACAS Xu, MNIST 和 CIFAR10 上将多路径回溯方法与使用单条回溯路径的 DeepPoly 进行定量比较, 在数据集 MNIST 上将多路径回溯方法与使用全局优化的 Optimized LiRPA 进行定量比较. 下面介绍实验使用的数据集和环境.

- ACAS Xu[20]网络用于小型无人机设备的防撞规避系统, 由 45 个全连接前馈 ReLU 网络组成. 这 45 个网络规模相同, 均为 6×50, 其中, 6 表示隐藏层层数, 50 表示每个隐藏层节点个数. 每个 ACAS Xu 网络输出层是 5 维, 分别表示网络给出的 5 种碰撞规避决策. 输入层是 5 维, 分别表示与其他飞行设备的距离及角度等 5 个参数, 是低维输入网络的代表.
- MNIST[21]是包含 0–9 这 10 个手写数字图像的数据集, 包含 6 万个训练样本以及 1 万个测试样本. 每个样本都是灰阶图像且大小均为 28×28 像素. 我们使用 MNIST 训练数据集训练了规模分别为 16×50, 10×80 及 20×50 的 3 个全连接前馈 ReLU 网络, 每个网络的输出层是 10 维, 分别表示网络给出的 10 个分类结果. 输入层为 28×28=784 维, 表示一张图片的 784 个灰度值, 是较高维输入网络的代表.
- CIFAR10[22]是包含 10 种物体图像的数据集, 包含 5 万个训练样本以及 1 万个测试样本. 每个样本都是 3 通道 RGB 图像, 且大小均为 32×32 像素. 我们使用 CIFAR10 训练数据集训练了规模分别为 10×100, 15×200 及 16×250 的 3 个全连接前馈 ReLU 网络, 每个网络的输出层是 10 维, 分别表示网络给出的 10 种分类结果. 输入层为 3×32×32=3072 维, 表示一张图片的 3 072 个灰度值, 是高维输入网络的代表.

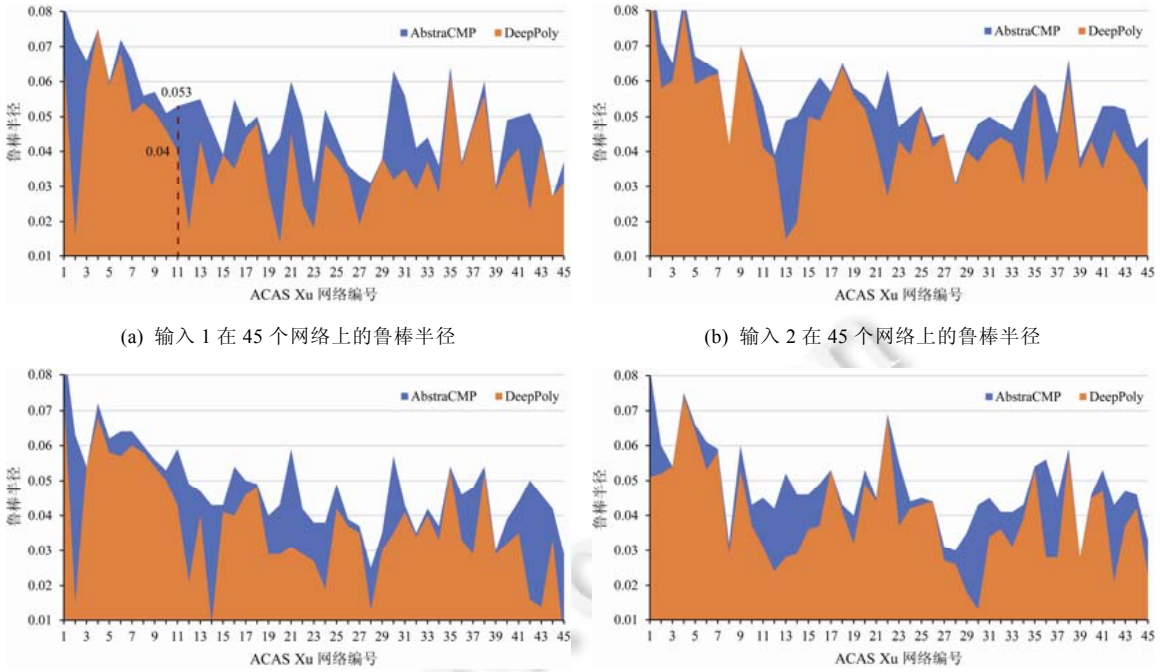
上述 ACAS Xu 网络和 MNIST 网络均使用归一化输入训练, 因此, 关于某个给定输入的扰动是归一化后的扰动, 关于给定输入的鲁棒半径也是归一化后的鲁棒半径.

我们使用的实验平台参数为: 处理器 Intel Core i7-6700@3.40 GHz; 内存大小 16 GB; 操作系统版本 Windows 10 专业版 64 位; Python 版本 3.9. 与 DeepPoly 的对比实验均在相同环境下进行.

3.1 低维输入网络上的精度改善

对于 ACAS Xu, 为了方便对比, 我们选择了 4 个随机合法输入, 分别通过 AbstraCMP 和 DeepPoly 计算这 4 个输入在 45 个网络上的鲁棒半径. 本实验中, 设定鲁棒半径精确到千分位; AbstraCMP 使用 M=3, 3 条回溯路径分别为 DeepPoly 路径、直角三角形抽象对应的回溯路径和钝角三角形抽象对应的回溯路径. 结果如图 6 所示.

图 6(a) 展示了对于输入 1, AbstraCMP 和 DeepPoly 在 45 个网络中能够验证的鲁棒半径. 其中, 深蓝色区域的上边界为 AbstraCMP 能够验证的鲁棒半径, 浅橙色为 DeepPoly 能够验证的鲁棒半径. 从图中红色虚线可以看出, 在第 11 个网络上, 两种方法得到的鲁棒半径分别为 0.053 和 0.04. 第 2 节中提到: 对于任意输入和任意网络, AbstraCMP 在理论上保证至少达到 DeepPoly 的效果. 因此, AbstraCMP 可验证的鲁棒半径(深蓝色区域)总是大于 DeepPoly 可验证的鲁棒半径(浅橙色区域). 通过图 6(b)–图 6(d)均能观察到同样的结论. 对于上近似方法, 同一输入下能验证更大的鲁棒半径, 意味着对真正可达集更精确的近似. 本实验中, AbstraCMP 能够验证的单个鲁棒半径最大比 DeepPoly 提高 370%; 对于每个输入, 在 45 个网络上能够验证的鲁棒半径平均值比 DeepPoly 提高 25–30%. 实验结果表明, AbstraCMP 在低维输入网络中的计算精度相比 DeepPoly 有所提升.



(a) 输入 1 在 45 个网络上的鲁棒半径

(b) 输入 2 在 45 个网络上的鲁棒半径

(c) 输入 3 在 45 个网络上的鲁棒半径

(d) 输入 4 在 45 个网络上的鲁棒半径

图 6 ACAS Xu 网络上能够验证的鲁棒半径对比

3.2 回溯路径数量对精度和时间的影响

在第 2 节提到: 理论上, 回溯路径可以任意多. 通过下面的实验, 我们将研究回溯路径数量的增加对鲁棒半径计算的改进以及带来的时间代价. 我们选定一个 ACAS Xu 网络, 并选定一个输入, 然后计算使用不同数量的回溯路径时, AbstraCMP 所能验证的鲁棒半径大小以及对应的时间开销.

关于回溯路径的选择, 首先, 我们选取 DeepPoly 路径作为 M=1 的配置; 此外, 由于定理 1 中任意$\lambda\in[0,1]$都构成对 ReLU 激活函数的可靠抽象, 所以我们选取[0,1]区间的等分值构成若干条回溯路径. 具体而言, M=2 时包含 DeepPoly 和λ=0 两条回溯路径; M=3 时包含 DeepPoly 和λ=0 及λ=1 共 3 条回溯路径; M=4 时包含 DeepPoly 及λ=0, 0.5, 1 共 4 条回溯路径; 以此类推. 本实验中设定鲁棒半径精确到千分位, 得到 AbstraCMP 能够验证的鲁棒半径随 M 值的变化如图 7 所示.

图 7 中, 带有三角形标签的蓝色实折线表示在给定网络和输入下, 随回溯路径数量 M 增加, AbstraCMP 能够验证的鲁棒半径变化. 橙色虚线表示 DeepPoly 在该网络和输入下能够验证的鲁棒半径, 虽然 DeepPoly 没有回溯路径的概念, 但为了方便对比, 我们将其能够验证的鲁棒半径 0.026 作为基准线. 从图 7 可以看出, 在 DeepPoly 的基础上, 增加较少的回溯路径就能取得相对于 DeepPoly 较好的效果. 在本实验的网络和输入下, 仅使用 M=2, 即可使计算得到的鲁棒半径相对于 DeepPoly 提高 57%.

M 值的增加可以获得更为精确的结果, 但这种改善并不能持续. 在该实验中, 我们发现: 随 M 值增加, AbstraCMP 能够验证的鲁棒半径会迅速收敛. 如图 7 所示, 当 M=7 时, 就已经达到收敛值 0.049. 其他实验表明, 对于大部分输入, 通常 M=3 或 M=4 已经接近收敛值. 因此, 为了平衡精度与时间, 我们认为 M=3 是一个较为合理的选择.

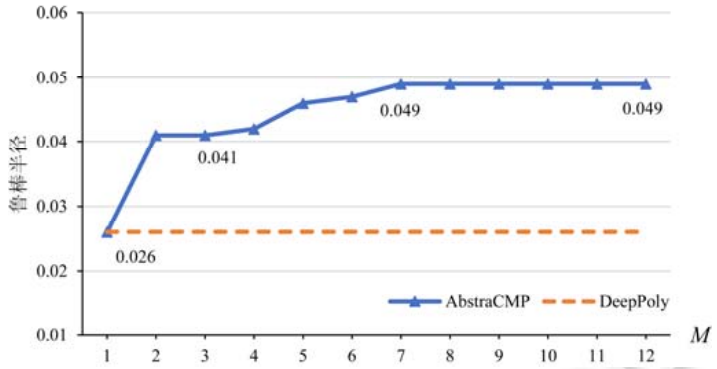


图 7 AbstraCMP 可验证鲁棒半径随 M 值的变化关系

表 1 给出了在上述网络和输入下, AbstraCMP 计算其鲁棒半径所需时间随回溯路径数量 M 的变化关系. 表中第 2 列是 DeepPoly 的用时, 第 3 列 M=2 及之后各列是取不同 M 值时 AbstraCMP 的用时. 对于每一列的配置, 实验记录其运行 10 次的平均时间. 正如第 2 节的讨论, AbstraCMP 的时间代价与回溯路径数量 M 线性相关, 从表 1 中也可以看出这一关系. DeepPoly 解决神经网络验证问题的时间代价是极低的, 当 M 取低值常数时, AbstraCMP 相较于现有验证方法而言时间代价也是极低的. 我们的方法所引入的额外时间代价很小.

表 1 AbstraCMP 验证时间随 M 值的变化关系

方法	DeepPoly	AbstraCMP								
		M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10
用时(s)	6.44	12.60	20.79	28.02	35.80	42.93	51.01	58.07	65.38	72.86

值得注意的是, 我们的方法在效率上存在较大的提升空间. AbstraCMP 目前虽然使用简单的单线程实现, 但多路径回溯方法在理论上可以高度并行化. 对于单条回溯路径, 同层各节点的数值上下界计算可以并行进行. 在文献[23]中, 作者介绍了 DeepPoly 的 GPU 实现. 而对于 AbstraCMP, 除同层各节点的并行以外, 同一节点不同回溯路径对应的数值上下界也可以并行计算, 以充分利用 CPU 甚至 GPU 的计算能力, 这将作为我们未来的一个工作方向.

3.3 高维输入网络上的精度改善

最后, 我们在数据集 MNIST 和 CIFAR10 上测试了 AbstraCMP 在高维输入网络上的表现. 对于 MNIST 数据集, 我们使用其训练集训练了 3 个全连接前馈神经网络, 它们的隐藏层规模分别为 16×50, 10×80 和 20×50. 我们选择测试集的前 100 张图片作为输入. 对比 AbstraCMP 和 DeepPoly 在不同扰动大小 θ 下能够成功验证鲁棒性的图片数量. 对于上近似方法, 给定一组输入和具体的 θ 值, 能够验证的鲁棒性越多, 自然说明这种方法越精确. 本实验中, AbstraCMP 取 M=3, 结果见表 2.

表 2 不同扰动下成功验证鲁棒性的图片数量(MNIST)

网络	方法	扰动大小 θ								
		0.010	0.012	0.015	0.017	0.020	0.022	0.025	0.027	0.030
MNIST 16×50	DeepPoly	89	82	71	66	52	40	26	23	16
	AbstraCMP	**90**	**86**	**76**	**70**	**58**	**45**	**28**	**23**	**18**
MNIST 10×80	DeepPoly	91	88	79	67	46	33	27	20	10
	AbstraCMP	**94**	**91**	**82**	**70**	**48**	**38**	**31**	**24**	**12**
MNIST 20×50	DeepPoly	73	70	49	40	31	22	16	13	7
	AbstraCMP	**80**	**72**	**58**	**49**	**37**	**27**	**20**	**18**	**10**

从表 2 中可以看到, 对于同一网络和相同的扰动大小, AbstraCMP 能够验证的图片数量总是大于 DeepPoly. 例如, 在网络 MNIST 20×50 上, 选定扰动大小 θ 为 0.010 时, DeepPoly 能够验证鲁棒的图片数量是 73, 而 AbstraCMP 能够验证 80 张图片的鲁棒性. 实验结果表明, AbstraCMP 在较高维输入网络上的计算精度

相比 DeepPoly 同样有所提升.

在本实验中, 对相同的扰动大小, AbstraCMP 能够成功验证的图片数量最多比 DeepPoly 多 42.9% (=(10−7)/7), 此情况出现在 MNIST 20×50 网络且扰动大小θ=0.030 时. 而且从实验结果可以观察到: 当网络隐藏层数量较多时, AbstraCMP 相比 DeepPoly 提升更加明显. 产生这一结果的原因是: 随着网络深度的增加, 多路径回溯方法在每个节点相较于单路径回溯提升的精度得到积累和放大.

对于 CIFAR10 数据集, 我们使用其训练集训练了隐藏层规模分别为 10×100, 15×200 及 16×250 的 3 个全连接前馈神经网络. 只有 ReLU 层的全连接 CIFAR10 网络相对于 MNIST 网络有较低的准确率和较小的鲁棒半径, 因此对于每个网络, 我们选择测试集分类正确的前 100 张图片作为输入, 对比 AbstraCMP 和 DeepPoly 在不同的扰动大小θ下, 能够成功验证鲁棒性的图片数量. 本实验中, AbstraCMP 取 M=3, 结果见表 3.

表 3 不同扰动下成功验证鲁棒性的图片数量(CIFAR10)

网络	方法	扰动大小θ								
		0.000 5	0.001 0	0.001 5	0.002 0	0.002 5	0.003 0	0.003 5	0.004 0	0.004 5
CIFAR10 10×100	DeepPoly	95	89	75	62	48	42	30	24	17
	AbstraCMP	**95**	**91**	**81**	**67**	**60**	**50**	**40**	**35**	**23**
CIFAR10 15×200	DeepPoly	93	87	75	64	57	48	35	32	21
	AbstraCMP	**94**	**88**	**79**	**70**	**61**	**55**	**47**	**35**	**29**
CIFAR10 16×250	DeepPoly	93	76	59	42	33	20	14	8	3
	AbstraCMP	**95**	**79**	**62**	**49**	**37**	**23**	**16**	**10**	**4**

在本实验中, 对相同的扰动大小, AbstraCMP 最多能够比 DeepPoly 多验证 12 张图片的鲁棒性, 此情况出现在扰动大小为 0.003 5 及网络规模为 15×200 时. 且在扰动大小为 0.003 5 时, AbstraCMP 在 3 个网络上平均比 DeepPoly 多验证 8 张图片的鲁棒性.

上述 3 组实验说明, 在不同规模的输入和网络下, AbstraCMP 的验证精度相较于 DeepPoly 均有所提升.

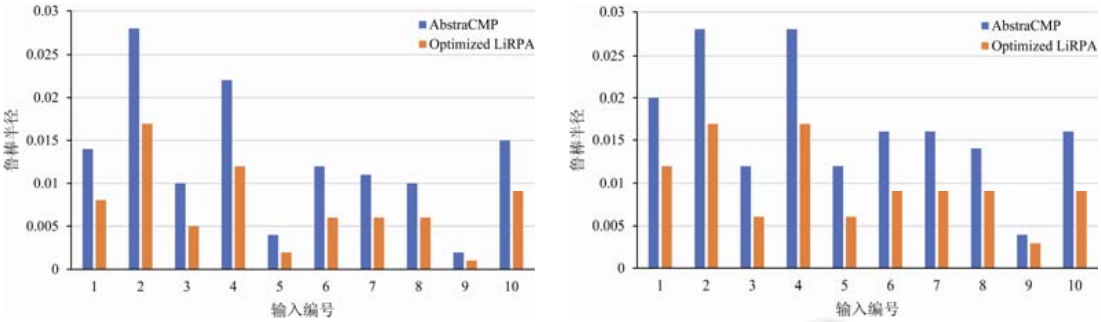
3.4 与Optimized LiRPA的精度比较

文献[24]提出了 Optimized LiRPA 工具, 与本文类似, 它对 ReLU 函数使用简单线性抽象; 不同的是, 它取 ReLU 抽象下界斜率λ为变量非具体值, 鲁棒性验证问题被转化为关于一组λ的非凸全局优化问题. 通过投影梯度下降等方法可以高效地求解.

虽然与本文使用不同的方法, 但 Optimized LiRPA 也是基于简单线性抽象的符号传播方法, 而且也是利用不同的λ以得到尽可能精确的验证结果. 因此我们认为, 有必要将本文方法与 Optimized LiRPA 在验证精度上相比较.

在本实验中, 我们取 MNIST 测试集的前 10 张图片作为输入, 分别使用 AbstraCMP 和 Optimized LiRPA 计算 10 个输入在两个 MNIST 网络上的鲁棒半径, 以比较两种方法的精度差异. 本实验中, 取 AbstraCMP 的 M=3; 使用 Optimized LiRPA 的原生实现且其梯度下降的迭代次数为 50 次; 设定鲁棒半径精确到千分位. 得到的结果如图 8 所示.

图 8 的柱形图中, 左边深蓝色柱形表示 AbstraCMP 在给定输入下得到的鲁棒半径, 相邻的浅橙色柱形表示 Optimized LiRPA 在给定输入下得到的鲁棒半径. 如图 8(a)中 MNIST 6×50 网络上, AbstraCMP 和 Optimized LiRPA 得到输入 1 的鲁棒半径分别为 0.014 和 0.008. 从图 8 可看出, 两种网络规模下, AbstraCMP 得到前 10 张图片的鲁棒半径均大于 Optimized LiRPA, 说明我们的方法相对于使用参数化λ的梯度下降方法仍然具有精度优势.



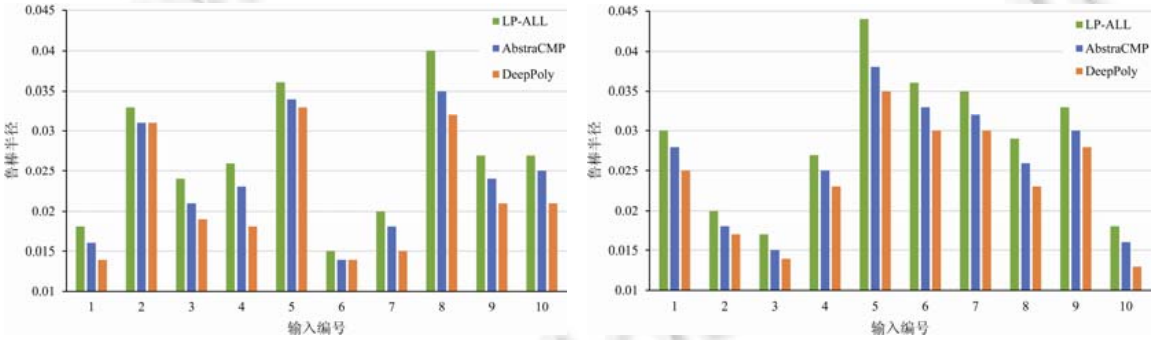
(a) MNIST 6×50 网络上的鲁棒半径对比　　(b) MNIST 8×80 网络上的鲁棒半径对比

图 8　AbstraCMP 与 Optimized LiRPA 能够验证的鲁棒半径对比

3.5　与LP-ALL的精度比较

文献[25]中, 作者 Salman 等人将单个神经网络节点的线性凸松弛方法统一为框架, 并说明这类方法存在精度上限. 对于 ReLU 网络, 文中将其精度上限对应的方法称为 LP-ALL. LP-ALL 逐层、逐节点构造 LP 问题以得到每个节点的数值上下界, 并使用得到的数值上下界构造 ReLU 抽象. 这是目前 ReLU 网络单个节点的线性凸松弛方法能够得到的最精确上下界. 另一方面, LP-ALL 需要解$\mathcal{O}(N)$个 LP 问题, 其中, N 为给定网络的节点数目, 且平均每个 LP 问题的变量数目为$\mathcal{O}(N)$个, 这使得目前 LP-ALL 对于较大规模网络是不太现实的方法. 文献[25]中使用 1 000 CPU 节点的集群计算了 MNIST 全体测试集图片对于给定扰动大小的鲁棒比例, 实验使用 1×500 和 2×100 两种规模的 MNIST 网络, 总计算量已经超过 22 CPU 年.

本节中, 我们将定量比较 LP-ALL, AbstraCMP 和 DeepPoly 的验证精度. 具体来说, 我们选择第 3.3 节中的两个网络: MNIST 10×80 和 MNIST 16×50, 每个网络分别选择 10 张图片作为输入, 比较 3 种方法得到这 10 个输入的鲁棒半径. 其中, AbstraCMP 使用 M=3, LP-ALL 求解器使用与文献[25]相同的 ECOS. 实验中设定鲁棒半径的精度阈值为千分位, 得到的结果如图 9 所示.



(a) MNIST 16×50 网络上的鲁棒半径对比　　(b) MNIST 10×80 网络上的鲁棒半径对比

图 9　LP-ALL, AbstraCMP, DeepPoly 能够验证的鲁棒半径对比

从图 9 中可看出, 正如 Salman 等人在文献[25]中提到的, LP-ALL 作为一种复杂度很高的方法, 却不能显著提升已有线性近似方法的验证精度. 对于本实验使用的全体输入集, LP-ALL 得到一个千分位鲁棒半径平均耗时 148×10^3 s (约 41 h), 但其精度相较 DeepPoly 仅平均绝对提升 0.005 0; 而 AbstraCMP 得到一个千分位鲁棒半径平均耗时 435 s, 其精度相较 DeepPoly 已具有 0.002 3 的平均绝对提升.

同时, 由图 9 也可知, AbstraCMP 在精度上弥补了 DeepPoly 和 LP-ALL 的间隙, 使得时间复杂度较低的回溯方法能够更加接近 LP-ALL 的验证效果.

4 相关工作

根据验证结果的保证程度划分, 本文提出的方法可靠但不完备. 对于ReLU网络, 可靠且完备的方法通常需要对激活状态不确定的ReLU节点分情况讨论. 按照实现技术不同, 这类方法可大致分为4组.

(1) 基于SMT. 在实现上, 将验证问题编码为可满足性问题来求解[7,14,26], ReLU函数可用SMT的if-then-else语句编码.

(2) 基于混合整数线性规划(MILP). 在实现上, 将验证问题编码为MILP问题, ReLU函数的激活与不激活状态分别对应整数决策变量的1和0, 从而可以使用Gurobi等支持MILP的求解器求解[27−29].

(3) 基于分支定界(BaB). 在实现上, 直接根据各种启发式策略来确定对哪个激活状态不确定的节点进行分支[24,30]. 另外, 目前也有工作使用图学习来指导节点的分支[31].

(4) 基于输入域精化. 通过反复分割输入域构成验证子问题, 在足够小的输入域下, 神经网络接近线性函数, 这使得可以分别验证这些子问题, 直到得到准确的验证结果或者超时[12].

前3组基于SMT, MILP以及BaB的方法本质上都是对激活状态不确定的节点进行分情况讨论, 所以最坏情况下, 需要描述能够确定该神经网络的2^N个线性函数, 其中, N为网络节点数量. 文献[7]也证明了其NP时间复杂度. 虽然已有很多高效的启发式策略来缓解这一复杂度, 但目前, 使用这类方法验证大规模网络仍然是不现实的.

对于使用线性抽象的方法, 通常越精确的抽象对真正可达集的近似程度越高. 值得注意的是, LP抽象虽然是单个ReLU函数的最精确线性抽象, 但并非对ReLU神经网络的最精确线性抽象, 它没有考虑同层多个节点取值的依赖关系. 利用这些依赖信息, 通常能够得到比LP更精确的抽象, 文献[27,32]都采用了这种想法.但另一方面, 通常越精确的抽象其时间代价也相对越高. 文献[33]中给出: 对于计算给定输入范围下神经网络的可达集这一问题, 除非$P=NP$, 否则不存在近似率为$(1-o(1))\ln N$的多项式时间算法, 其中, N为网络节点数量.

文献[25]将单个神经网络节点的线性凸松弛方法统一为框架, 并从理论和实验上说明单个节点的线性凸松弛方法与未松弛的原网络验证问题之间有难以弥补的间隙, 作者称这一间隙为convex barrier. 本文基于简单线性抽象的符号传播方法在其框架中属于greedy algorithm的范畴, 在文献[34]中被称为界限传播(bound propagation)方法. 但在多路径回溯的概念下, 目前这类方法均使用单条回溯路径, 因此仅能得到每个节点的一个上下界. 使用多条回溯路径可以得到每个节点的多个上下界, 从而提升验证精度.

目前已有一些工作改进基于线性抽象的符号传播方法的精度, 如RefineZono[16]选择性地使用LP甚至MILP来得到部分节点更精确的上下界以及DeepSRGR[17]将LP和DeepPoly相结合迭代地进行验证. 这些方法都能有效提高验证精度, 但它们都不同程度地引入LP问题的求解, 具有较高的时间代价. 相比之下, 本文是完全基于简单线性抽象的符号传播方法, 在引入较低时间代价的同时, 可获得较高的精度提升. 另外, 本文的方法也可以结合LP等更加复杂但精度更高的方法, 从而进一步提高验证精度.

5 结 论

在对神经网络上近似处理的验证方法中, 基于线性抽象的符号传播方法具有很强的代表性, 它在精度与时间之间进行了折中: 一方面, 它不像区间算术一样十分不精确, 以至于难以验证有意义的性质; 另一方面, 也不像LP一样十分耗时, 以至于难以处理较大规模的网络. 本文提出了多路径回溯的概念,将目前基于简单线性抽象的符号传播方法从单条回溯路径扩展到多条回溯路径, 并分析了多条回溯路径的可靠性和复杂度. 实验结果也表明, 我们的方法能够明显提高验证精度, 而仅引入较小的额外时间代价.

值得注意的是, 虽然本文仅关注激活函数为ReLU的神经网络验证问题, 但我们的方法不仅适用于ReLU函数, 同样适用于其他能够被线性抽象的激活函数, 如tanh和sigmoid等, 这将作为我们接下来的一个扩展方向. 另一方面, 尽管我们的方法可以取任意多条回溯路径, 但每条路径使用单一的抽象方式. 如果能够在单条回溯路径上有效地选择不同的抽象方式, 理论上精度能够获得进一步的提升. 使用全局优化方法的Optimized

LiRPA 实际上提供了找到一组较好λ的思路. 类似的想法还有文献[35], 作者把验证问题对应的两阶段凸优化问题重构为非凸问题, 从而能够求解更大规模的网络. 最优化方法对于神经网络验证具有重要影响, 因为这一问题本身就对应于优化问题, 未来我们也将会探讨本文方法与优化方法的联系.

References:

[1] Dong YP, Su H, Zhu J. Towards interpretable deep neural networks by leveraging adversarial examples. Acta Automatica Sinica, 2020, 48(1): 75−86 (in Chinese with English abstract).

[2] Huang X, Kwiatkowska M, Wang S, Wu M. Safety verification of deep neural networks. In: Proc. of the 29th Int'l Conf. on Computer Aided Verification. LNCS 10426, Cham: Springer, 2017. 3−29.

[3] Wang Z, Yan M, Liu S, Chen JJ, Zhang DD, Wu Z, Chen X. Survey on testing of deep neural networks. Ruan Jian Xue Bao/Journal of Software, 2020, 31(5): 1255−1275 (in Chinese with English abstract). http://www. jos.org.cn/1000-9825/5951.htm [doi: 10.13328/j.cnki.jos.005951]

[4] Liu C, 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.

[5] Li L, Qi X, Xie T, Li B. SoK: Certified robustness for deep neural networks. CoRR abs/2009.04131, 2020.

[6] Huang X, Kroening D, Ruan W, Sharp J, Sun Y, Thamo E, Wu M, Yi X. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review, 2020, 37: Article No.100270.

[7] 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. LNCS 10426, Cham: Springer, 2017. 97−117.

[8] Singh G, Gehr T, Mirman M, Püschel M, Vechev M. Fast and effective robustness certification. In: Advances in Neural Information Processing Systems 31: Annual Conf. on Neural Information Processing Systems. 2018. 10825−10836.

[9] Tran HD, Manzanas Lopez D, Musau P, Yang X, Nguyen LV, Xiang W, Johnson TT. Star-based reachability analysis for deep neural networks. In: Proc. of the 23rd Int'l Symp. on Formal Methods (FM 2019). LNCS 11800, Cham: Springer, 2019. 670−686.

[10] Raghunathan A, Steinhardt J, Liang P. Semidefinite relaxations for certifying robustness to adversarial examples. In: Advances in Neural Information Processing Systems 31: Annual Conf. on Neural Information Processing Systems. 2018. 10900−10910.

[11] Dathathri S, Dvijotham K, Kurakin A, Raghunathan A, Uesato J, Bunel R, Shankar S, Steinhardt J, Goodfellow IJ, Liang P, Kohli P. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In: Advances in Neural Information Processing Systems 33: Annual Conf. on Neural Information Processing Systems. 2020.

[12] Wang S, Pei K, Whitehouse J, Yang J, Jana S. Formal security analysis of neural networks using symbolic intervals. In: Proc. of the 27th USENIX Security Symp. (USENIX Security 2018). 2018. 1599−1614.

[13] 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): Article No.41.

[14] Ehlers R. Formal verification of piece-wise linear feed-forward neural networks. In: D'Souza D, Narayan Kumar K, eds. Proc. of the 15th Automated Technology for Verification and Analysis (ATVA 2017). LNCS 10482, Cham: Springer, 2017. 269−286.

[15] Wang S, Pei K, Whitehouse J, Yang J, Jana S. Efficient formal safety analysis of neural networks. In: Advances in Neural Information Processing Systems 31. 2018. 6369−6379.

[16] Singh G, Gehr T, Püschel M, Vechev M. Robustness certification with refinement. In: Proc. of the Int'l Conf. on Learning Representations. 2019.

[17] Yang P, Li R, Li J, Huang CC, Wang J, Sun J, Xue B, Zhang L. Improving neural network verification through spurious region guided refinement. In: Proc. of the 27th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Vol.12651. 2021. 389−408.

[18] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proc. of the 2017 IEEE Symp. on Security and Privacy (SP). 2017. 39−57.

[19] 2021. https://github.com/zhengyeah/abstracmp

[20] Julian KD, Lopez J, Brush JS, Owen MP, Kochenderfer MJ. Policy compression for aircraft collision avoidance systems. In: Proc. of the 35th IEEE/AIAA Digital Avionics Systems Conf. (DASC). 2016. 1−10.

[21] 2021. http://yann.lecun.com/exdb/mnist/

[22] 2021. https://www.cs.toronto.edu/~kriz/cifar.html

[23] Müller C, Serre F, Singh G, Püschel M, Vechev M. Scaling polyhedral neural network verification on GPUs. Proc. of Machine Learning and Systems, 2021, 3: 733−746.

[24] Xu K, Zhang H, Wang S, Wang Y, Jana S, Lin X, Hsieh CJ. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In: Proc. of the 9th Int'l Conf. on Learning Representations. 2021.

[25] Salman H, Yang G, Zhang H, Hsieh CJ, Zhang P. A convex relaxation barrier to tight robustness verification of neural networks. In: Advances in Neural Information Processing Systems 32. 2019. 9832−9842.

[26] Katz G, Huang DA, Ibeling D, Julian K, Lazarus C, Lim R, Shah P, Thakoor S, Wu H, Zeljić A, Dill DL, Kochenderfer MJ, Barrett C. The marabou framework for verification and analysis of deep neural networks. In: Proc. of the Int'l Conf. on Computer Aided Verification. LNCS 11561, Cham: Springer, 2019. 443−452.

[27] Tjeng V, Xiao K, Tedrake R. Evaluating robustness of neural networks with mixed integer programming. arXiv: 1711.07356, 2019.

[28] Anderson R, Huchette J, Ma W, Tjandraatmadja C, Vielma JP. Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming, 2020, 183(1-2): 3−39.

[29] Dutta S, Jha S, Sankaranarayanan S, Tiwari A. Output range analysis for deep feedforward neural networks. In: Proc. of the 10th Int'l Symp. on NASA Formal Methods, Vol.10811. 2018. 121−138.

[30] Bunel R, Lu J, Turkaslan I, Torr PHS, Kohli P, Kumar MP. Branch and bound for piecewise linear neural network verification. Journal of Machine Learning Research, 2020, 21(42): 1−39.

[31] Lu J, Kumar MP. Neural network branching for neural network verification. In: Proc. of the 8th Int'l Conf. on Learning Representations. 2020.

[32] Singh G, Ganvir R, Püschel M, Vechev M. Beyond the single neuron convex barrier for neural network certification. In: Advances in Neural Information Processing Systems 32. 2019.

[33] Weng L, Zhang H, Chen H, Song Z, Hsieh CJ, Daniel L, Boning D, Dhillon I. Towards fast computation of certified robustness for relu networks. In: Proc. of the 35th Int'l Conf. on Machine Learning, Vol.80. 2018. 5276−5285.

[34] Xu K, Shi Z, Zhang H, Wang Y, Chang KW, Huang M, Kailkhura B, Lin X, Hsieh CJ. Automatic perturbation analysis for scalable certified robustness and beyond. In: Advances in Neural Information Processing Systems 33: Annual Conf. on Neural Information Processing Systems. 2020. 1129−1141.

[35] Bunel R, Hinder O, Bhojanapalli S, Dvijotham K. An efficient nonconvex reformulation of stagewise convex optimization problems. In: Advances in Neural Information Processing Systems 33: Annual Conf. on Neural Information Processing Systems. 2020.

附中文参考文献:

[1] 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. 自动化学报, 2020, 48(1): 75−86.

[3] 王赞, 闫明, 刘爽, 陈俊洁, 张栋迪, 吴卓, 陈翔. 深度神经网络测试研究综述. 软件学报, 2020, 31(5): 1255−1275. http://www.jos.org.cn/1000-9825/5951.htm [doi: 10.13328/j.cnki.jos.005951]

郑烨(1998－), 男, 硕士生, CCF 学生会员, 主要研究领域为智能系统的安全性, 神经网络验证.

刘嘉祥(1987－), 男, 博士, 助理教授, CCF 专业会员, 主要研究领域为形式化方法, 程序验证, 神经网络验证.

施晓牧(1987－), 女, 博士, 研究员, CCF 专业会员, 主要研究领域为形式化方法, 定理证明技术及其在安全关键系统的应用.