Model Robustness

> 由于我学习模型鲁棒性时，从一开始就是学习的 SOK，因此，该模块的大部分内容均会在 SOK 那篇文章中进行展开，其他文章的阅读仅是作为一个扩展。

Model Robustness

- Todo List
- Overview
- Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks
  - Contribution
  - Notes
    - Linear Approximation
      - $ReLU$
  - Links
- Certified Defenses For Adversarial Patches
  - Contribution
  - Notes
  - Questions
  - Links
- SoK: Certified Robustness for Deep Neural Networks
  - Contribution
  - Notes
    - Notations and Preliminaries
      - Adversarial Perturbation
      - Optimization Problem
      - Input & Ouput
      - Linear Programming
      - Stable & Unstable $ReLU$
    - Deterministic Approaches
      - Branch & Bound
    - Probabilistic Approaches
  - Links

Todo List

1. Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In International Conference on Machine Learning, pp. 3575–3583, 2018.
2. Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. arXiv preprint arXiv:1810.12715, 2018.
3. 解决 Certified Defenses For Adversarial Patches 中的问题；

Overview

开始 ”模型鲁棒性“ 这个板块之前，问几个问题：

1. 我们的预期是什么？
2. 已有的工作模式？
3. 如何来评估一个工作的好坏？
4. 里面的困难有哪些？

Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks

Contribution

Notes

Linear Approximation

考虑前馈神经网络中的 $ReLU$ 单元和 $MaxPool$ 单元具有非线性，所以期望用线性约束来替换掉这两个单元。

$ReLU$

Links

- 论文链接：Ehlers R. Formal verification of piece-wise linear feed-forward neural networks[C]//International Symposium on Automated Technology for Verification and Analysis. Springer, Cham, 2017: 269-286.

Certified Defenses For Adversarial Patches

Contribution

1. 文章的亮点在于：将 **IBP（Interval Bound Propagation）** 这种模型鲁棒性方法运用到了防御 **带patch的对抗攻击**中；
2. 有趣的点：文章发现这样训练的模型对不同 patch 的对抗攻击都具有一定的鲁棒性；

Notes

1. Introduction: 图像存在对抗攻击, 实际物理对抗攻击都是带 patch 的攻击, 所以作者想做带 patch 的鲁棒性检测;
2. Problem Setup:

   (1) 定义 **模型的鲁棒性**:

$$\mathop{\mathbb{E}}_{x\sim X} \min_{p\in\mathbb{P},l\in\mathbb{L}} \mathcal{X}[f(A(x,p,l);\theta)=y]$$

其中, $\mathbb{L}$ 表示 patch 的位置集合, $\mathbb{P}$ 表示特定的 patch 集合, X 表示样本集合, $\mathcal{X}[\circ]$ 是布尔函数 (若为真返回1, 若为假返回0). 公式的含义是在输入集合上**无法**得到成功的对抗样本的输入数量, 对于攻击者而言, 这个值越小越好, 而对于防御者而言, 这个值越大越好;

(2) 定义patch: 正方形的 patch, 其中的值可以为任意值 [0~1];

3. Vulnerability of Existing Defenses: 作者借此说明现有的一些依靠经验性结果的对抗样本防御方法 (**针对那些带 patch 的无目标攻击**) 很容易被攻击者绕过.

    - 已有的防御方法:

      (1) 防御原理: **输入图像的损失函数的梯度值在添加对抗扰动的地方都会很大** . Watermarking 方法基于这个原理来防御无目标攻击 (值得关注的是, 这种方法对模型的成功率的影响约为 12%) ;

      (2) 防御原理: **输入图像的像素值在这些添加了对抗扰动的地方会变化很大, 即不连续** . Local Gradient Smoothing 方法基于这个原理, 对输入作预处理;

      $$\hat{x} = x \odot (1 - \lambda g(x))$$

    - 已有的防御方法容易被绕过:

		Patch Size		
Attack	Defense	42×42	52×52	60×60
IFGSM	LGS	78%	75%	71%
IFGSM + LGS	LGS	14%	5%	3%
IFGSM	DW	56%	49%	45%
IFGSM + DW	DW	13%	8%	5%

    在生成对抗样本的过程中即引入防御算法, 就可以绕过这些防御方法.

4. Certified Defenses:

(1) 在输入的 $p - norm$ 邻域内, 检查模型的分类结果是否会发生改变, 如果改变, 则不鲁棒, 可能是一个对抗样本; 如果不改变, 则鲁棒, 不可能是一个对抗样本;

(2) 检查输入是否是鲁棒的是十分苦难的, **因为计算复杂性高, 是一个 NP-Hard 问题**;

(3) 鲁棒性的边界是十分宽松的, **即只有在很小一个邻域内, 才能保证模型的鲁棒性**;

(4) Interval Bound Propagation (IBP) 的原理: 想象一下, 你有一些**相互独立的**且已知取值区间的变量 x, y, z , 有一个这些变量组成的线性表达式 $f = ax + by + cz$, 现在你想求这个表达式的取值区间, 那么很简单, 你让表达式的每一项最大即可得到最大值, 让每一项最小即可得到最小值. (这种方法是一种放缩求解的方法, 因为给定了**变量相互独立**这个条件, 但实际神经网络中, 中间层的神经元的取值之间存在某种线性关系) 看具体的公式:

- 只考虑仿射变换的情况下:

$$\overline{z}^{(k)} = W^{(k)} \frac{\overline{z}^{(k-1)} + \underline{z}^{(k-1)}}{2} + |W^{(k)}| \frac{\overline{z}^{(k-1)} - \underline{z}^{(k-1)}}{2} + b^{(k)}$$
$$\underline{z}^{(k)} = W^{(k)} \frac{\overline{z}^{(k-1)} + \underline{z}^{(k-1)}}{2} - |W^{(k)}| \frac{\overline{z}^{(k-1)} - \underline{z}^{(k-1)}}{2} + b^{(k)}$$

  这个公式不太好看, 自己去推一下, 结果是这样的:

$$\overline{z}^{k} = W^{(k)+} \overline{z}^{(k-1)} + W^{(k)-} \underline{z}^{(k-1)} + b^{(k)}$$

  其中 $W^{(k)+}$ 是将 $W^{(k)}$ 中小于 0 的部分置为0, $W^{(k)-}$ 将大于0的部分置为0, 这样就和上面原理部分相对应;

- 考虑激活函数: 那就在外面套个激活函数, 考虑一下激活函数的**单调性**即可;
- 鲁棒性验证: 最后每一个分类都可以得到一个取值区间, 保证目标分类的最小值 大于其他分类的最大值即可, 作者用如下公式

$$\underline{\mathbf{m}}_y = e_{y_{true}}^T \underline{z}^{(K)} - e_y^T \overline{z}^{(K)} = \underline{z}_{y_{true}}^{(K)} - \overline{z}_y^{(K)} \geq 0 \quad \forall y$$

- 鲁棒性训练, 来增强 IBP 的鲁棒性:

- 修改 loss 函数, 从原来的希望目标分类尽可能大, 到添加扰动后的概率区间更满足鲁棒性 (默认包含了分类正确):

$$\text{Certificate Loss} = \text{Cross Entropy Loss}(-\underline{\mathbf{m}}, y)$$

- 训练的 trick: (**todo: 没看懂**)

(5) ☆ 作者提出的方法: 在 IBP 的基础上, 把扰动的区间限制在一个**矩形范围**内, 并把扰动的大小限制在 $[0, 1]$. 但是 IBP 方法自身的复杂性，再加上需要考虑每一个位置的 Patch，**计算复杂性过大**，所以作者提出了**两种挑选 Patch 位置的方法**；

- 基本框架：选择patch的形状大小后，在图像的各个位置上进行遍历，分析最坏情况；
  - 鲁棒性验证：

$$\underline{\mathbf{m}}^{\text{es}}(\mathbb{L})_y = \min_{l \in \mathbb{L}} \underline{\mathbf{m}}^{\text{single patch}}(l)_y \quad \forall y.$$

  - 鲁棒性训练：

$$\text{Certificate Loss} = \text{Cross Entropy Loss}(-\underline{\mathbf{m}}^{\text{es}}(\mathbb{L}), y).$$

    其中 $\mathbb{L}$ 表示 图片中放patch位置的集合；
- Random Patch Certificate Training: **随机选择一些 patch 的位置**, 随机选择一些可能的 patch 取值;
  - 鲁棒性验证：

$$\underline{\mathbf{m}}^{\text{random patches}}(\mathbb{L})_y = \underline{\mathbf{m}}^{\text{es}}(S)_y$$

  - 鲁棒性训练：

$$\text{Random Patch Certificate Loss} = \text{Cross Entropy Loss}(-\underline{\mathbf{m}}^{\text{random patches}}(\mathbb{L}), y)$$

- Guided Patch Certificate Training：使用 **U-Net 网络**为每个分类挑选一个 patch；
- Defense against Sparse Attack：作者指出，可以将 IBP 第一层的区间公式改成如下形式 （作者直接放这个式子真的是玄学，猜了半天猜出大概的思路，因为 Sparse Attack 就是将几个离散的对模型结果影响最大的像素点的值（值域为 $0 \sim 1$）进行修改，而不同像素点对结果的影响主要还是从权重矩阵中体现出来，所以**会修改权重矩阵中的** $top-k$ **。至于为什么可以直接 加/减** $top-k$ **呢**？因为是值域限定了，我最多也就是把 $0 \to 1$，或者是 $1 \to 0$，所以这个区间一定是满足全部的可能性的。至于**为什么没有了偏置项** b **呢**？我觉得很可能是两种写法，一种是将偏置项单独写出来，另一种写法是将偏置项包含在了权重项 W 中 ）

$$\overline{z}_i^{(1)} = W_{i,:}^{(1)} z^{(0)} + |W_{i,:}^{(1)}|_{top_k} \qquad \underline{z}_i^{(1)} = W_{i,:}^{(1)} z^{(0)} - |W_{i,:}^{(1)}|_{top_k} \qquad \forall i$$

5. 实验结果：

(1) 鲁棒性验证——防御攻击的效果：

Table 2: Comparison of our IBP certified patch defense against existing defenses. Empirical adversarial accuracy is calculated for 400 random images in both datasets. All results are averaged over three different models.

Dataset	Patch Size	Adversary	Defense	Clean Accuracy	Empirical Adversarial Accuracy	Certified Accuracy
MNIST	2×2	IFGSM	None	98.4%	80.1%	-
	2×2	IFGSM	LGS	97.4%	90.0%	-
	2×2	IFGSM + LGS	LGS	97.4%	60.7%	-
	2×2	IFGSM	IBP	98.5%	93.9%	91.6%
	5×5	IFGSM	None	98.5%	3.3%	-
	5×5	IFGSM	IBP	92.9%	66.1%	62.0%
CIFAR	2×2	IFGSM	None	66.3%	25.4%	-
	2×2	IFGSM	LGS	64.9%	31.3%	-
	2×2	IFGSM + LGS	LGS	64.9%	24.2%	-
	2×2	IFGSM	DW	47.1%	43.3%	-
	2×2	IFGSM + DW	DW	47.1%	20.2%	-
	2×2	IFGSM	IBP	48.6%	45.2%	41.6%
	5×5	IFGSM	None	66.5%	0.4%	-
	5×5	IFGSM	LGS	51.2%	22.11%	-
	5×5	IFGSM + LGS	LGS	51.2%	0.5%	-
	5×5	IFGSM	DW	45.3%	59.3%	-
	5×5	IFGSM + DW	DW	45.3%	15.6%	-
	5×5	IFGSM	IBP	33.9%	29.1%	24.9%

(2) 鲁棒性训练——防御攻击的效果：使用的 Patch 越多，放置的位置越多，得到的模型效果越好；

Table 3: Trade-off between certified accuracy and training time for different strategies. The numbers next to training strategies indicate the number of patches used for estimating the lower bound during training. Most training times are measured on a single 2080Ti GPU, with the exception of all-patch training which is run on four 2080Ti GPUs. For that specific case, the training time is multiplied by 4 for fair comparison. See Appendix A.6 for more detailed statistics. *indicates the performance of the best performing large model trained with either random or guided patch. Detailed performance of the large models can be found in Appendix A.5

		2×2			5×5		
Dataset	Training Strategy	Clean Accuracy	Certified Accuracy	Training Time(h)	Clean Accuracy	Certified Accuracy	Training Time(h)
MNIST	All Patch	98.5%	91.5%	9.3	92.0%	60.4%	8.4
	Random(1)	98.5%	82.9%	0.2	96.9%	24.1%	0.4
	Random(5)	98.6%	86.6%	0.3	95.8%	42.1%	0.3
	Random(10)	98.6%	87.7%	0.3	95.6%	49.6%	0.3
	Guided(10)	98.6%	88.9%	2.2	95.0%	53.1%	2.6
CIFAR	All Patch	50.9%	39.9%	56.4	33.5%	22.0%	45.8
	Random(1)	53.6%	21.6%	0.6	43.6%	6.1%	0.6
	Random(5)	52.9%	32.3%	0.7	39.0%	14.6%	0.7
	Random(10)	51.9%	35.6%	0.8	38.8%	18.6%	0.8
	Guided(10)	52.4%	36.0%	3.7	37.9%	18.8%	3.7
	Large Model*	**65.8%**	**51.9%**	22.4	**47.8%**	**30.3%**	15.4

(3) 对于 Sparse Attack 的防御效果：

Table 4: Certified accuracy for sparse defenses with IBP and Random Ablation.

Dataset	Sparsity (k)	Model	Clean Accuracy	Certified Accuracy
MNIST	1	IBP-sparse	98.4%	96.0%
	4	IBP-sparse	97.8%	90.8%
	10	IBP-sparse	95.2%	86.8%
	1	Random Ablation	96.7%	90.3%
	4	Random Ablation	96.7%	79.1%
	10	Random Ablation	96.7%	29.2%
CIFAR	1	IBP-sparse	48.4%	40.0%
	4	IBP-sparse	42.2%	31.2%
	10	IBP-sparse	37.0%	25.6%
	1	Random Ablation	78.3%	68.6%
	4	Random Ablation	78.3%	61.3%
	10	Random Ablation	78.3%	45.0%

(4) 用矩形 Patch 训练的模型能否防御其他形状的 Patch？**这个还挺有趣的，方法是存在一定的 Transferability 的**；

Table 5: Certified accuracy for square-patch trained model for different shapes

Dataset	Pixel Count	Square	Rectangle	Line	Diamond	Parallelogram
MNIST	4	91.6%	-	92.5%	91.6%	92.3%
	16	69.4%	55.4%	46.7%	68.13%	70.2%
	25	59.7%	50.9%	32.4%	53.6%	55.2%
CIFAR	4	50.8%	-	46.1%	48.6%	49.8%
	16	36.9%	29.0%	32.1%	35.7%	36.3%
	25	30.3%	25.1%	29.0%	30.1%	30.7%

Questions

1. 文章中提到的 U-Net 网络怎么用在实际的工作中?
2. 模型鲁棒性验证法是如何进行实验的?
3. 实验结果中的 Certified Accuracy 是如何计算出来的?
4. 鲁棒性训练里面的Trick没有看懂?

Links

- 论文链接: Chiang, Ping-yeh, et al. "Certified defenses for adversarial patches." *ICLR (2020).*
- 源码链接: Ping-C / certifiedpatchdefense
- $p-norm$ 详解: 知乎 / 0范数, 1 范数, 2范数有什么区别?
- U-Net 网络详解: 图像语义分割入门+FCN/U-Net网络解析

SoK: Certified Robustness for Deep Neural Networks

Contribution

Notes

Notations and Preliminaries

Adversarial Perturbation

定义在输入 x_0 处的扰动边界:

$$B_{p,\epsilon}(x_0) := \{x : \|x - x_0\|_p \leq \epsilon\}$$

Optimization Problem

模型鲁棒性的证明问题，可以转化为最优化问题:

$$\mathcal{M}(y_0, y_t) = \min_x f_\theta(x)_{y_0} - f_\theta(x)_{y_t} \quad s.t. \, x \in B_{p,\epsilon}(x_0)$$
$$\text{Goal : To Prove} \quad \forall y_t, \mathcal{M}(y_0, y_t) \geq 0$$

即，我们希望在 对抗扰动 的最坏情况下，模型还能将样本分类为正确的标签。

Input & Ouput

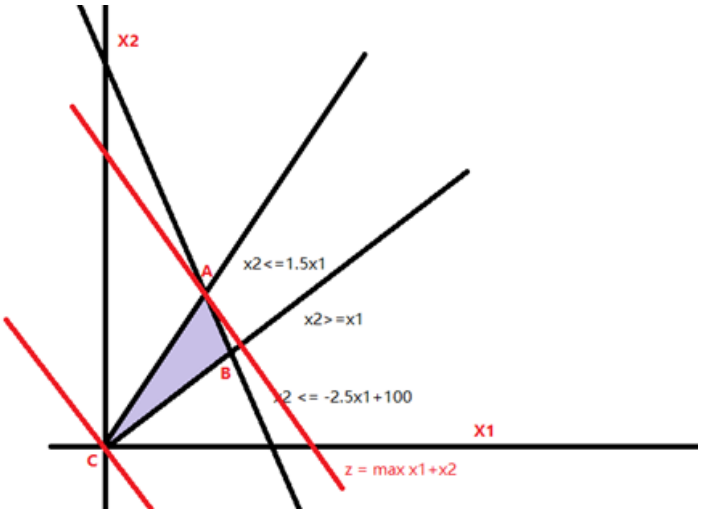
文章中我们讨论神经元的输入输出时，我们**将单层神经网路的线性计算和激活函数拆开讨论。**因为线性计算本身具有线性性质，易于推导其输出值；而激活函数则大多都是非线性的，故不易推导其输出值；

Linear Programming

线性规划问题，在一系列线性约束下，求线性目标函数的最值：

$$\min_x c^T x \quad s.t. \begin{cases} Ax \leq b \\ x \geq 0 \end{cases}$$

可以看到，**线性规划问题和鲁棒性证明问题之间是非常像的。**举例如下问题即为一个线性规划问题：



那么如何来求解线性规划问题？大致的做法是首先找到一个初始点（这个点是边界的某个角），然后让其在边界上滑动。具体参考：

- 参考链接一：https://www.jianshu.com/p/a0fc8a57f452
- 参考链接二：https://brilliant.org/wiki/linear-programming

Stable & Unstable $ReLU$

对于一个 $ReLU$ 单元，其输入为 $\hat{x}$：

- 若 $\hat{x}$ 始终为正或始终为负，则我们称这个 $ReLU$ 是 `Stable`；
- 反之，若 $\hat{x}$ 的值域在 0 的两侧，则我们称这个 $ReLU$ 是 `Unstable`；

Deterministic Approaches

> 论述顺序和原文会有很大的区别，原文是为了更好地去分类这些方法，而我追求的是方法上面的连贯性。

确定性证明方法，直接确定 `模型在输入点处是鲁棒的` 这个命题，形式化为：$\mathcal{A}(f_\theta, x_0, y_0, \epsilon) = False$ 。

Branch & Bound

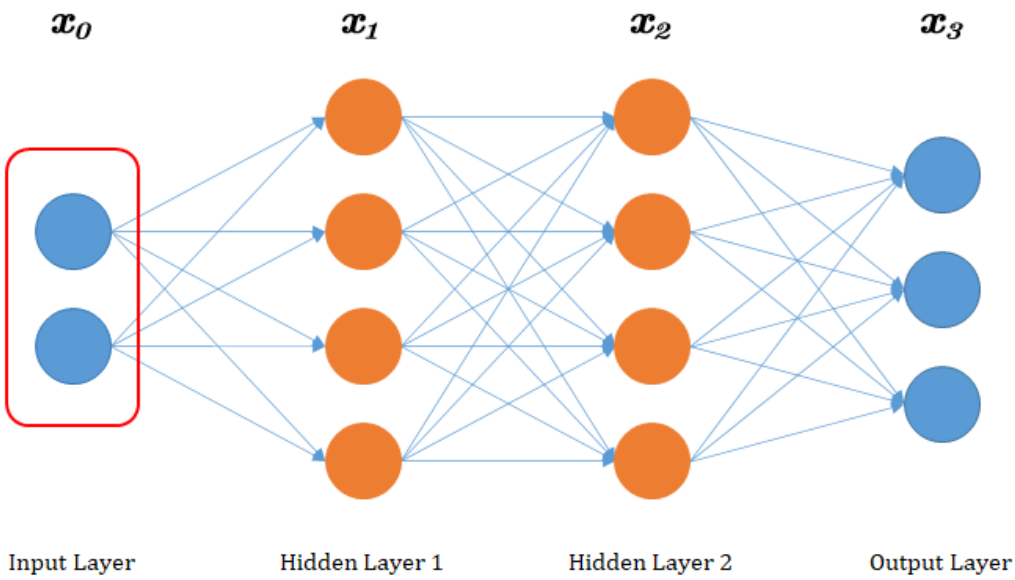
> 这个讲解过程遵循 “提出问题-解决问题”的思路，过程中会有一些疑惑，比如说 最大值/最小值怎么去求解？计算的前后关系是怎样的？所以现在这边说明一下，可以先不要计较这些问题，而关注每个小节的重点问题，如：
>
> - 第一小节，你应该关注如何将问题转化为一个线性规划问题；

> - 第二小节，你应该先关注如何对神经元值域进行放缩，再关注如何在前后层神经元之间进行值域的传递；
> - 第三小节，你应该关注为什么我们又将这个问题返回到了原始的线性规划问题，整个方法这样做有什么意义；
>
> 最后我根据给老师讲解的时候收集到的反馈，在 PPT 之外补充了 “第四小节-计算过程”，来厘清这个方法具体是如何进行计算的；

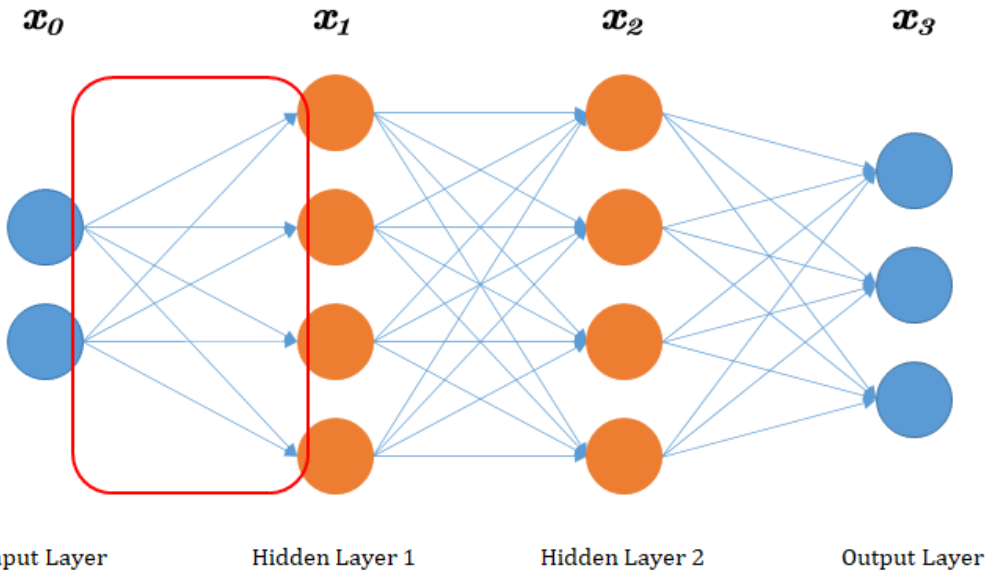
1. **Linear Programming**

   首先考虑将鲁棒性证明问题转化为一个线性规划问题。

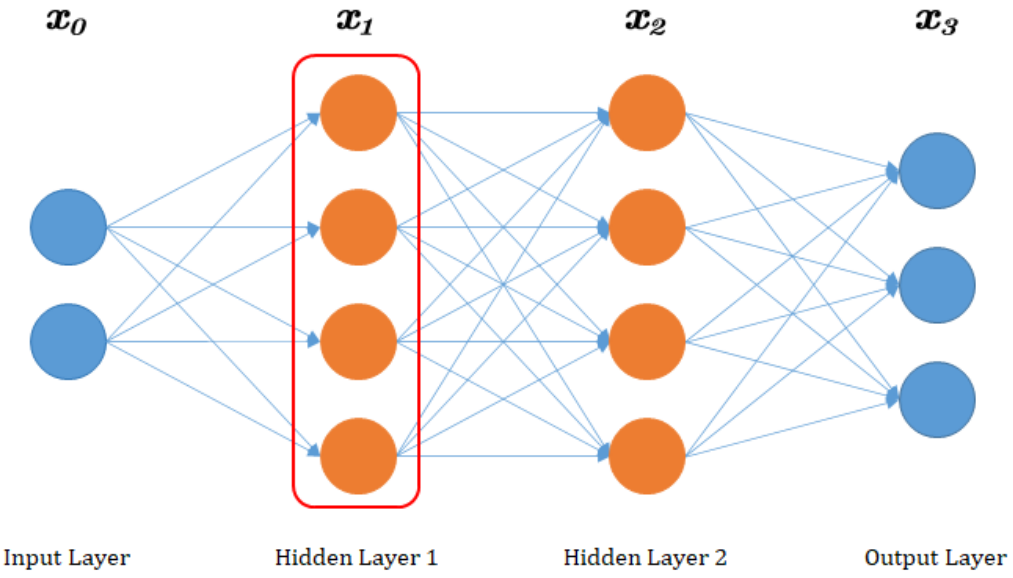
   (1) 输入约束 - 不等式约束：



   (2) 线性计算 - 等式约束：



   (3) 激活函数 - 不等式 + 等式约束：



(4) 线性规划问题：

现在假设 $x_{1,0}$ 神经元是一个 $ReLU$ 单元，那么我们**根据该神经元输入值的大小进行分类讨论**：

$$\min_{\boldsymbol{x}} f_\theta(\boldsymbol{x})_{y_0} - f_\theta(\boldsymbol{x})_{y_t} \qquad \min_{\boldsymbol{x}} f_\theta(\boldsymbol{x})_{y_0} - f_\theta(\boldsymbol{x})_{y_t}$$

$$\begin{cases} \text{constraint } 1 \\ \text{constraint } 2 \\ \hat{x}_{1,0} \le 0 \\ x_{1,0} = 0 \\ \cdots\cdots \end{cases} \qquad \begin{cases} \text{constraint } 1 \\ \text{constraint } 2 \\ \hat{x}_{1,0} \ge 0 \\ x_{1,0} = \hat{x}_{1,0} \\ \cdots\cdots \end{cases}$$

那么如果神经网络中有 n 个 $ReLU$ 神经元，我们就需要有 2^n 个线性规划问题需要求解，**计算复杂度十分高**，故**我们要提出一种方法来解决计算复杂性问题。**

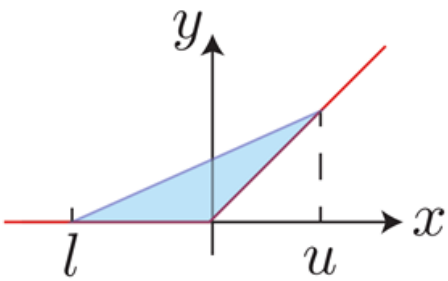
2. **Linear Relaxation**

线性放缩，在其他文章中也称为线性逼近，即**用线性边界来拟合神经网络中的非线性单元的边界。**

(1) $ReLu$ 单元

原文中，称线性放缩后的 $ReLU$ 单元为 $ReLU\ Polytope$ 。

对于 $ReLU$ 单元，其本身是非线性的，是一个分段线性函数（分为大于 0，和小于 0 两端）：



现在假设我们知道 $ReLU$ **输入**的最小值 $l_{i,j}$ 和最大值 $u_{i,j}$，并且最小值小于 0 而最大值大于 0：

$$l_{i,j} \le \min_{x \in B_{p,\epsilon}(x_0)} \hat{x}_{i,j} \le \max_{x \in B_{p,\epsilon}(x_0)} \hat{x}_{i,j} \le u_{i,j}$$
$$l_{i,j} < 0 < u_{i,j}$$

那么我们就可以用三个线性约束来确定$ReLU$ **输出**的值域：

$$
\begin{gathered}
x_{i,j} \geq 0 \\
x_{i,j} \geq \hat{x}_{i,j} \\
x_{i,j} \leq \frac{u_{i,j}}{u_{i,j}-l_{i,j}}(\hat{x}_{i,j}-l_{i,j})
\end{gathered}
$$

现在我们可以将上面 2^n 个线性规划问题，转换成 1 个线性规划问题：

$$
\begin{gathered}
\min_{\boldsymbol{x}} f_\theta(\boldsymbol{x})_{y_0} - f_\theta(\boldsymbol{x})_{y_t} \\
\begin{cases}
\text{constraint } 1 \\
\text{constraint } 2 \\
x_{1,0} \geq 0 \\
x_{1,0} \geq \hat{x}_{1,0} \\
x_{1,0} \leq \frac{u_{i,j}}{u_{i,j}-l_{i,j}}(\hat{x}_{i,j}-l_{i,j}) \\
\cdots\cdots
\end{cases}
\end{gathered}
$$

(2) $MaxPool$ 单元

对于 $MaxPool$ 单元，由于它是比较多个值之间的最大值，故形式上会显得更复杂，且不太容易理解。为了更好地解释，定义 d 是$MaxPool$ 单元的输出，而 $c_k \ \ (k \in \{1,\ldots,k\})$ 是 $MaxPool$ 的 k 个输入。

首先，我们可以简单地列出如下线性约束：

$$
\begin{gathered}
\bigwedge_{i\in\{1,\ldots,k\}} (d \geq c_i) \\
\max_{i\in\{1,\ldots,k\}} c_i \geq d
\end{gathered}
$$

但是，在线性规划问题中，并没有 max 这个约束，所以需要做一些转换，下面直接给出 文章 中的式子，然后给出解释：

$$
\left[\bigwedge_{i\in\{1,\ldots,k\}} (d \geq c_i)\right] \quad \wedge \quad \left[c_1+\cdots+c_k \geq d+(\sum_{i\in\{1,\ldots,k\}} l_i) - \max_{i\in\{1,\ldots,k\}} l_i\right]
$$

左边的式子是显然可得的，我们需要思考一下右边这个式子，假设 $l_n = \max_{i\in\{1,\ldots,k\}} l_i$ ，我们可以将右边的不等式写成如下形式：

$$
(c_1-l_1)+\cdots+\cancel{(c_n-l_n)}+\cdots+(c_k-l_k)+c_n \geq d
$$

- 如果恰好 $c_n = \max_{i\in\{1,\ldots,k\}} l_i$ ； 由于 $(c_i - l_i) \geq 0$，所以上式成立；
- 如果 $c_j = \max_{i\in\{1,\ldots,k\}} l_i$ 并且 $j \neq n$ ； 则我们知道 $c_j \geq c_n \geq l_n \geq l_j$，所以我们可以得到如下式子：

  $$
  (c_j-l_j)+c_n = (c_j - c_n + c_n - l_n + l_n - l_j)+c_n = (c_n-l_n)+(l_n-l_j)+c_j
  $$

  那么我们可以改写一下上面的式子：

  $$
  (c_1-l_1)+\cdots+\cancel{(c_j-l_j)}+\cdots+(c_n-l_n)+\cdots+(c_k-l_k)+(l_n-l_j)+c_j \geq d
  $$

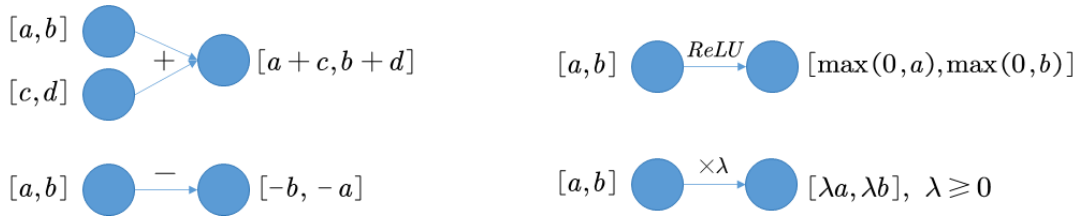
  由于我们已知 $l_n \geq l_j$ ，所以上式成立；

现在我们可以将上面的 2^n 个线性规划问题，转换成 1 个线性规划问题：（这里不再对公式的小标作更严谨的讨论）

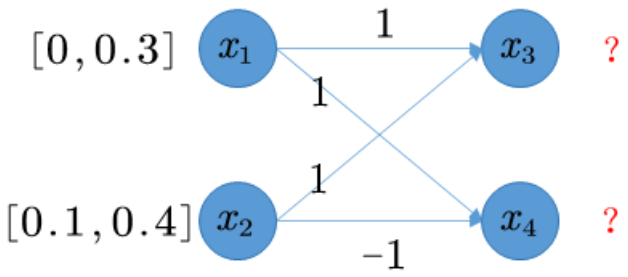
$$
\begin{gathered}
\min_{\boldsymbol{x}} f_\theta(\boldsymbol{x})_{y_0} - f_\theta(\boldsymbol{x})_{y_t} \\
\begin{cases}
\text{constraint } 1 \\
\text{constraint } 2 \\
d \geq c_i \\
c_1+\cdots+c_k \geq d+(\sum_{i\in\{1,\ldots,k\}} l_i) - \max_{i\in\{1,\ldots,k\}} l_i \\
\cdots\cdots
\end{cases}
\end{gathered}
$$

(3) 神经元输入的取值区间

确定输出的取值范围之前，我们仍不知道输入的范围 $[l_{i,j},\ u_{i,j}]$，这里可以通**过放缩的方式通根据区间范围来求输入的范围**：



举例来说，我们可以用上述方法来求下图中的 x_3, x_4：



$$
\begin{aligned}
x_3 &= x_1 + x_2 \\
x_4 &= x_1 - x_2 \\
0 &\leqslant x_1 \leqslant 0.3 \\
0.1 &\leqslant x_2 \leqslant 0.4
\end{aligned}
$$

$$
\begin{aligned}
0.1 &\leqslant x_3 \leqslant 0.7 \\
-0.4 &\leqslant x_4 \leqslant 0.2
\end{aligned}
$$

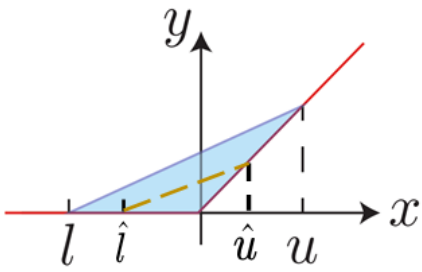
可以看到，这里实际上是**忽略了输入之间存在的线性关系。**

3. **Divide & Conquer**

(1) Un-Complete：至此，上述通过求解 1 个线性规划问题的方法，是一种 Un-Complete（不完备）方法。假设我们求得 $\mathcal{M} = \min_x f_\theta(x)_{y_0} - f_\theta(x)_{y_t}$：

- 如果 $\mathcal{M} \geq 0$，那么我们可以直接给出结论：模型 f_θ 在输入 x 的 ϵ 扰动邻域内是鲁棒的；
- 如果 $\mathcal{M} < 0$，此时我们并**不能直接给出 "不鲁棒" 的结论**；

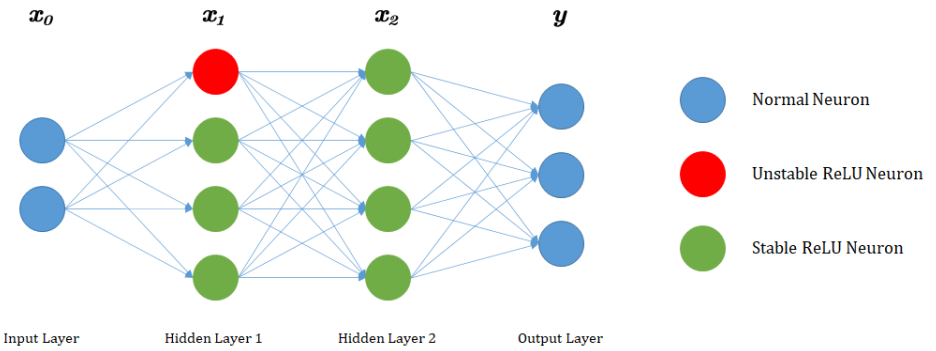
  以放缩后的 $ReLU$ 单元为例：



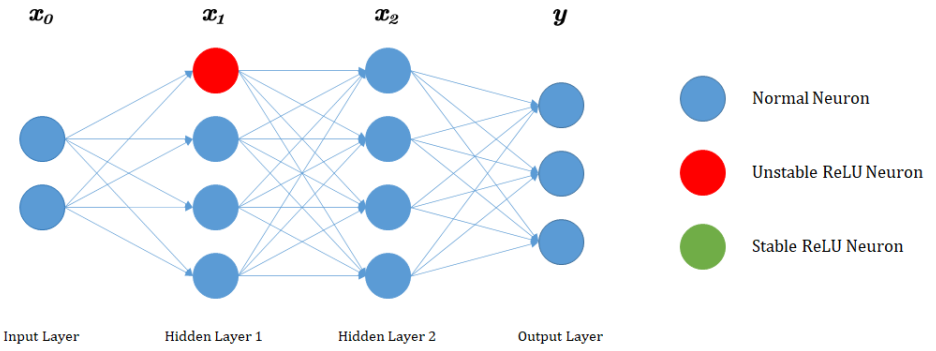
图中，$\hat{l}$ 和 $\hat{u}$ 为输入的实际上下界，而 l 和 u 则为我们用区间范围估计得到的输入上下界。现在，当我们取 $\mathcal{M} < 0$ 时，取值点并非在红色折现上（实际的 $ReLU$ 单元取值区间），而是在蓝色三角形区域（放缩后的 $ReLU\ Polytope$ 取值区间）。

(2) 替换网络中 $Unstable$ 的 $ReLu$

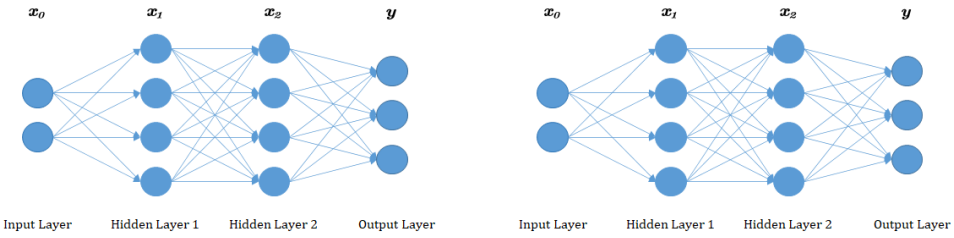
- 假设我们现有如下图所示的神经网络，蓝色表示正常线性神经元，红色为 $Unstable$ 的 $ReLU$ 单元，绿色为 $Stable$ 的 $ReLU$ 单元：



- 我们可以将其中的 $Stable$ 的 $ReLU$ 单元直接替换成正常的线性神经元：



- 最后，我们可以根据 $Unstable$ 的 $ReLU$ 单元进行分类讨论，再将 1 个线性规划问题，拆分成 2 个线性规划问题：（可以看到，到这里其实我们又回到了最原始的 2^n 个线性规划问题的版本，但是注意，只有我们通过求解 1 个线性规划问题无法得出模型鲁棒的结论时，才需要再进行这步计算）



$$\min_{\boldsymbol{x}} f_\theta(\boldsymbol{x})_{y_0} - f_\theta(\boldsymbol{x})_{y_t}$$
$$\begin{cases} \text{constraint 1} \\ \text{constraint 2} \\ \hat{x}_{1,0} \leqslant 0 \\ x_{1,0} = 0 \\ \cdots\cdots \end{cases}$$

$$\min_{\boldsymbol{x}} f_\theta(\boldsymbol{x})_{y_0} - f_\theta(\boldsymbol{x})_{y_t}$$
$$\begin{cases} \text{constraint 1} \\ \text{constraint 2} \\ \hat{x}_{1,0} \geqslant 0 \\ x_{1,0} = \hat{x}_{1,0} \\ \cdots\cdots \end{cases}$$

(3) Complete：因为我们已经回到了原始的线性规划问题上来，故该方法是 Complete （完备）的；

4. **计算流程**

Probabilistic Approaches

概率性证明方法，证明 `模型在输入点处是鲁棒的概率大于阈值` 这个命题，形式化为：$\Pr[\mathcal{A}(f_\theta, x_0, y_0, \epsilon)] \geq 1 - \alpha$ 。

Links

- 论文链接：Li L, Qi X, Xie T, et al. SoK: Certified Robustness for Deep Neural Networks[J]. arXiv preprint arXiv:2009.04131, 2020.
- 论文代码：VeriGauge